Eötvös Loránd Tudományegyetem

Természettudományi Kar

Kis-Benedek Ágnes

Dinamikus programozás a gráfelméletben

BSc szakdolgozat

Témavezető:

Frank András, egyetemi tanár

Operációkutatási Tanszék

Budapest, 2010

Előszó

Szeretnék ezúton is köszönetet mondani témavezetőmnek, Frank Andrásnak a rengeteg segítségért, időért és figyelemért, és hogy sosem jöhettem el tőle új gondolkodnivaló nélkül.

Tartalomjegyzék

Bevezető

A szakdolgozat célja a dinamikus programozás nevű technika bemutatása, egy jól érthető, magyar nyelvű gyűjtemény létrehozása annak gráfelmélettel kapcsolatos alkalmazásairól, valamint a problémák hasonlóságának, kapcsolatainak, esetleges visszavezethetőségeinek vizsgálata. A szakdolgozatban kiemelt szerepet kap a hasonló struktúrájú (elsősorban maximális lánc, illetve minimális út keresésére visszavezethető) feladatok rendszerbe foglalása.

A problémák és algoritmusok gyűjtése során az adott probléma fontossága, illetve a dinamikus programozás különböző vonásainak bemutathatósága jelentették a fő szempontokat. Az első fejezetben viszonylag egyszerűbb, de fontos gráfelméleti problémákkal foglalkozunk (felhasznált irodalom: [4], [6]). A második fejezetbe került feladatok jóval komplexebbek, ezen kívül van köztük rögzítetlen paraméterre NP-teljes feladat (felhasznált irodalom: [1], [5], [8], [9]). A harmadik fejezet már a maximális lánc keresésének problémájára visszavezethető, hasonló struktúrájú feladatokkal foglalkozik (felhasznált irodalom: [1], [2]). A negyedik fejezetben látszólag nagyon különböző problémák kapcsolatainak vizsgálata történik (felhasznált irodalom: [2], [3], [4], [6]). Egyes algoritmusok Frank András ötletei, illetve a sajátjaim nyomán kerültek kidolgozásra. (Pl.: 1.4, 2.3, 3.5, 4.1, 4.7 alfejezetek, illetve a visszavezetések egy része.) Más algoritmusokat az irodalomjegyzékben szereplő szakirodalmak alapján dolgoztam fel, kiszűrve a szakdolgozat szempontjából fontos információkat. (Pszeudokódokat, részletesebb leírásokat, egyéb, a szakdolgozat kereteibe nem illeszkedő további problémákat és algoritmusokat az érdeklődő azokban találhat.)

A dinamikus programozást gyakran optimalizálási feladatok megoldására hasz-

nálják. Számos olyan feladat van, amelyet megoldhatunk divide et impera módszerrel is "valahogy", de a dinamikus programozás hatékonyan is képes minderre, vagy a mohó stratégia nem bizonyul megfelelőnek, míg a dinamikus programozás megtalálja a keresett optimumot.

A dinamikus programozás lényege, hogy az eredeti feladat megoldása helyett sok új alproblémát határozunk meg, melyek piramisszerűen egymásra épülnek, és így a legegyszerűbbek megoldása szinte triviális, továbbá egy új alprobléma megoldását az előzőek ismeretében könnyen elő tudjuk állítani. Ha az összes ilyen általunk generált nagy mennyiségű alproblémát megoldjuk, az eredeti feladat megoldását is megkapjuk, vagyis bár a feladatok száma jelentősen növekszik, és bizonyos értelemben sokkal többet oldunk meg az eredeti problémánál, a feladatok egymásra épülése és finomabb különbsége miatt ezen problémák megoldásai között már könnyű az átjárás. Például az 1. fejezetben található első útkeresési algoritmusnál ahelyett, hogy egy minimális st-út költségét határoznánk meg, ahogy a feladat szól, bevezetünk sok új alproblémát, mégpedig minden v pontra megadjuk a maximum i élű ($i = 0, ..., n-1$) minimális vt-utak költségét. Bár ez jóval több probléma, $i = 0$-ra a megoldás triviális, ha pedig már valamilyen i-re ismerjük az összes alprobléma megoldását, az $i+1$ eset könnyen számítható.

Általában, hogy ez az alproblémákkal bővítés megvalósítható legyen, teljesülnie kell a következőknek: az optimális megoldás egy része optimális megoldását adja a megfelelő alproblémának, és az alproblémák optimális megoldásaiból felépíthető az eredeti feladat megoldása. Továbbá fontos, hogy az alproblémák száma ne legyen túl nagy, valamint polinomiális időben megoldhatók legyenek, és ezek megoldásainak ismeretében hatékonyan számíthassuk ki az eredeti feladat optimumát.

Minden optimalizálási feladathoz rendelhető egy fastruktúra, melynek neve döntési fa. A gyökérpont jelöli az eredeti feladathoz tartozó állapotot, az első szinten elhelyezkedő csomópontok azokat az állapotokat, melyekbe a feladat az első döntés után átkerülhet, stb. A cél egy optimális döntéssorozat meghatározása. A feladat egy ilyen döntés nyomán hasonló típusú feladatra redukálódhat, vagy kisebb alfeladatokra bomolhat. Bár egy döntési fában exponenciálisan sok levél keletkezhet, a csomópontok között találhatunk azonosakat, az ezek által képviselt feladatokat

nyilván csak egyszer érdemes megoldani. Az összevont döntési fa - ami már tulajdonképp nem fa, hanem egy irányított gráf -, ezen azonos csomópontok összevonásával kapható.

A konkrét algoritmikus megvalósításnál tehát egyrészt egy a feltételeknek megfelelő rekurziót kell felírni, ez adja meg, miként épülnek fel az egyszerűbb feladatok optimumából a bonyolultabb feladatok optimumai, és ez magában foglalja az optimális döntéshozás módját is, másrészt fontos, hogy a részeredményeket csak egyszer számoljuk ki, utána tároljuk őket. Gyakran célszerű magukat az optimális választásokat is eltárolni, hogy gyorsabban rekonstruálhassuk az optimális döntéssorozatot.

Fontos hangsúlyozni, hogy az alfeladatok optimális megoldásából nem építhetjük fel tetszőlegesen az eredeti optimumot. Például 13 forint kifizetését szeretnénk megoldani minél kevesebb érme segítségével, ha tetszőleges számú 1 forintos, 2 forintos, 5 forintos, illetve 10 forintos érménk van. Bár a feladat felbontható $6 + 7$ forint kifizetésére, azok optimumai $5 + 1$, illetve $5 + 2$, ami együtt négy darab érme, míg a 13 forint kifizetése $10+2+1$ módon három érmével is megoldható. Éppen a megfelelő alfeladatokra bontás az a döntéssorozat, amit meg kell meghatározni.

A módszer jelentőségét jól mutatja, hogy NP-teljes problémáknál a paramétert rögzítettnek tekintve (FTP - fixed parameter tractability) gyakran létezik hatékony dinamikus programozási algoritmus a megoldás megtalálására. (Például ilyen a 2. fejezetben található Steiner-fákra vonatkozó algoritmus.)

1. fejezet

Útkeresési problémák

Ebben a fejezetben fontos útkeresési és részgráfokkal kapcsolatos problémákról lesz szó.

1.1. Minimális st-út keresése

Az úgynevezett Bellman-Ford algoritmus segítségével oldjuk meg a problémát. A dinamikus programozás, mint általános programozási technika Bellman nevéhez kötődik, az 1950-es években dolgozta ki. A minimális út keresése ennek a technikának egyik első alkalmazása.

Feladat: Adott egy $G = (V, E)$ irányított gráf, élein c konzervatív költségfüggvény, valamint egy $s, t \in V$ pontpár. A cél minimális költségű st-út meghatározása.

Ehelyett kicsit bővebb feladatot oldunk meg: minden v pontra meg fogjuk határozni a v-ből t-be vezető minimális i-élű utak költségét, ahol $0 \leq i \leq n-1$.

Állítás: Ha $P[uv]$ az u és v pontok közötti egyik legrövidebb út, $x \in P[uv]$, akkor a $P[uv]$ út u-tól x-ig vezető szakasza, illetve a $P[uv]$ út x-től v-ig vezető szakasza külön-külön is optimális.

Állítás: Ha G-ben nincs negatív kör, egy minimális st-út maximum $n-1$ élből állhat. (Ez a részfeladatok számának szempontjából érdekes.)

Jelölje $opt(i, v)$ a maximum i db élt használó minimális vt-utak költségét. Célunk meghatározni $opt(n-1, v)$ értékét.

Világos, hogy $opt(0, v) = \infty$, kivéve $opt(0, t) = 0$-t. A további részmegoldások kiszámítására a következő rekurziót írhatjuk fel:

$$opt(i, v) = min\{opt(i-1, v), min\{opt(i-1, w) + c(vw)\}\}$$

ahol $vw \in E$, $0 < i$. Ez az összefüggés abból következik, hogy egy v-ből t-be vezető maximum i élű séta vagy i-nél kevesebb élből áll, és akkor már megtaláltuk az optimumot, vagy először egy vw-él mentén eljutunk egy w pontba, és onnan legfeljebb $i-1$ él felhasználásával megyünk tovább a t pontba.

Az algoritmus futásideje $O(|V||E|)$, mivel adott i-re a maximum i élű séták optimumainak meghatározásánál $O(|E|)$ műveletet kell végzünk, és ezt minden $i = 1, \ldots, n-1$-re meg kell tennünk.

1.2. Negatív kör létezése

Érdekes kérdés lehet az is, hogy mi történik, ha nem tudjuk, hogy létezik-e negatív kör a gráfban. A Bellman-Ford algoritmusra vonatkozó néhány megfigyelés segít ennek eldöntésében.

Adott G gráfhoz készítsük el a következő bővített G' gráfot: legyen t egy új pont, és minden G-beli pontból vegyünk fel egy 0 súlyú, t-be mutató élt.

Állítás: G'-ben létezik negatív kör úttal t-be $\Leftrightarrow$ G-ben létezik negatív kör.

Bizonyítás: Ha G-ben létezik negatív kör, akkor G'-ben is létezik, hiszen minden G-beli élt és pontot megtartottunk, és mivel minden pontból vezet él t-be, ezért a negatív körből is. Ha pedig G'-ben létezik negatív kör, akkor az nem tartalmazhatja t-t, mert t-ből nem vezet ki él, ez pedig csak úgy lehetséges, ha t-t elhagyva is létezik

benne negatív kör, ekkor pedig éppen G-t kapjuk vissza.

Állítás: Ha létezik vt-séta, és tartalmaz egy C negatív kört, akkor ezen a körön tetszőlegesen sokszor végigmenve folyamatosan csökken a célfüggvény értéke, vagyis $\lim_{i\to\infty} opt(i,v) = -\infty$.

Állítás: Ha nincs negatív kör a gráfban, $opt(i,v) = opt(n-1,v)\ \forall v\ \forall i \geq n$.

Állítás: A gráfban nem létezik C negatív összköltségű kör $\Leftrightarrow opt(n,v) = opt(n-1,v)\ \forall v \in V$.

Bizonyítás: Ha nincs negatív kör a gráfban, minden minimális út maximum $n-1$ élből áll, így $opt(n,v) = opt(n-1,v)$ nyilvánvalóan teljesül. Ha pedig $opt(n,v) = opt(n-1,v)$, akkor $opt(n+1,v)$ értéke is ugyanennyi lesz, ugyanis azt a rekurzió során az $opt(n,v)$-k segítségével számoljuk, amik viszont megegyeznek az $opt(n-1,v)$-kel, s így a rekurzió ugyanazt adja $opt(n+1,v)$-re, mint amit $opt(n,v)$-re is adott. Ugyanígy $opt(n-1,v) = opt(n,v) = opt(n+1,v) = opt(n+2,v) = \ldots$. Ebből pedig következik, hogy $\lim_{i\to\infty} opt(i,v) \neq -\infty$, vagyis a gráf nem tartalmaz negatív kört.

Tehát ha ki szeretnénk deríteni, hogy egy G gráf tartalmaz-e negatív kört, először megkonstruáljuk a G' gráfot, majd erre alkalmazhatjuk a Bellman-Ford algoritmust t-be vezető minimális utak meghatározására, de $n-1$ helyett n-ig kiszámítva az $opt(i,v)$-k értékét, és a végén ellenőrizve, hogy megfelelő eredményeket kaptunk-e.

1.3. Minimális st-út minden pontpárra

Feladat: Adott egy $G = (V,E)$ irányított gráf, élein c konzervatív költségfüggvénnyel. A cél minimális költségű st-út meghatározása minden $s,t \in V$ pontpárra.

A feladat megoldásához a Floyd-Warshall algoritmust fogjuk használni.

Jelölje $d_{ij}(k)$ az olyan minimális ij-út költségét, ahol teljesül az a kikötés, hogy minden belső pont az első k db pont közül kerül ki. $k = 0$ esetén nyilván teljesül

a $d_{ij}(k) = c_{ij}$ összefüggés. Ha $k \geq 1$, akkor az első k pontot használó minimális út vagy megegyezik az első $k-1$ pontot használóval, vagy az út során pontosan egyszer felhasználjuk a k-adik pontot is. (Kétszer biztosan nem térünk oda vissza, mivel a gráf nem tartalmaz negatív kört.) Ebből a megfigyelésből adódik a következő rekurzió a $k \geq 1$ esetre:

$d_{ij}(k) = min\{d_{ij}(k-1), d_{ik}(k-1) + d_{kj}(k-1)\}$.

Az algoritmus egy lépése során kiszámoljuk $d_{ij}(k)$-t $\forall\, i, j \in V$-re, ezt folytatjuk k-t 0-tól n-ig, és így megkapjuk a belső pontként csak az első k pontot használó utak értékét minden pontpárra, és egyre nagyobb k-kra. $k = n$ esetén végül megkapjuk a feladat megoldását, mivel itt tulajdonképpen már nincs semmilyen megkötés az út belső pontjaira vonatkozólag.

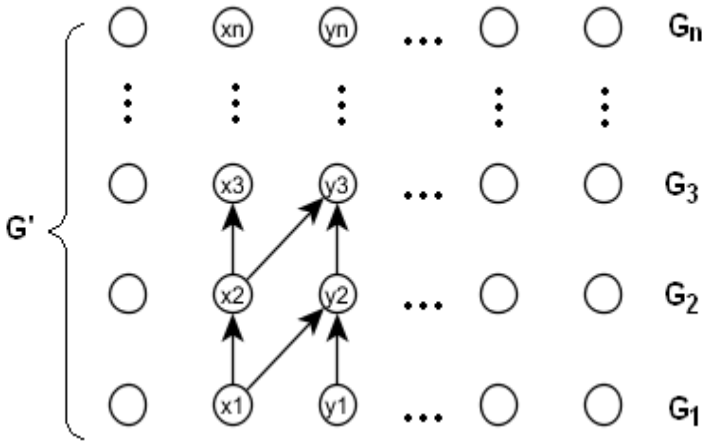
Az algoritmus futásideje $O(n^3)$, mivel minden ij pontpárra ($O(n^2)$), valamint minden k-ra ($k = 1, \ldots, n$) meg kell határoznunk $d_{ij}(k)$-t, egy adott $d_{ij}(k)$ értéke viszont az előzőek ismeretében már konstans sok művelettel elvégezhető.

1.4. "Felfújt" konstrukció

Egy tetszőleges $G = (V, E)$ irányított gráfhoz elkészíthető egy aciklikus $G' = (V', E')$ bővített gráf, mely n darab pontosztályból áll, vagyis $V' = V_1 \cup \cdots \cup V_n$, ahol $V_i = V$, az élhalmaza pedig a következőképp néz ki: $x_i y_{i+1} \in E'$, $i \in \{1, \ldots, n-1\}$, ha $xy \in E$, azaz egy pontosztály valamely pontjából a nála eggyel nagyobb sorszámú pontosztály valamely pontjába pontosan akkor megy él, ha az eredeti G gráfban a nekik megfelelő pontok közt vezetett él.

Az így konstruált G' gráf aciklikus lesz, mert az egyes pontosztályokon belül nem vezet él, és egy pontosztályból csak nála nagyobb sorszámú pontosztályokba juthatunk el. Egy topologikus sorrendet adnak a pontok pontosztályonként növekvő sorrendben felsorolva, tehát először V_1, majd $V_2, \ldots$, végül V_n pontjai.

Aciklikus digráfban adott s pontból a többi pontba úgy kaphatjuk meg a legrövidebb utak egy F fenyőjét, hogy vesszük a $\mu(v_i)$ függvényt, amely a maximum i élű séták minimális költségét jelöli, és ezt a pontok topologikus sorredjében dolgozva egy egyszerű rekurzió segítségével kaphatjuk $j > i$-re: $\mu(vj) = min\{\mu(v_i) + c(v_i v_j) :$



$v_iv_j \in E\}$. Amelyik élen a minimum felvétetik, azt hozzávesszük F-hez.

s-t jelen esetben válasszuk G_1-ből. Ekkor a fenti topologikus sorrendben számolva μ értékét, tulajdonképpen szintenként haladunk, vagyis meghatározzuk s-ből a G_2-beli pontokba vezető minimális utat, majd a G_3-beliekbe, G_4-beliekbe, stb.

Ha az eredeti gráfban veszünk egy maximum i élű sétát ($i \leq n$), ebben a G' gráfban találunk neki megfelelő sétát, ha ugyanazokat az éleket használjuk, csak itt mindig eggyel nagyobb sorszámú pontosztályba lépve. Sőt, a G-beli maximum i élű séták megfelelnek G'-ben az első i pontosztályt használó sétáknak.

Ez azt jelenti, hogy az első alfejezetben látott módszer, amely a séták élszámával dolgozik, megfeleltethető egy nagyobb méretű, aciklikus digráf minimális útjainak meghatározásával (a μ függvényt használva). G'-ben a két módszer teljesen hasonlóan működik.

A módosított G' gráfban G aciklikusságát is ellenőrizhetjük, hiszen G pontosan akkor aciklikus, ha a G'-beli minimális x_1y_{n-1}-út hossza megegyezik a minimális x_1y_n-út hosszával, minden x-re és y-ra.

1.5. Maximális pozitív, adott r pontot tartalmazó részfa

Feladat: Adott egy $T = (V, E)$ fagráf, $r \in V$ gyökérpont, c_v a csúcsokon értelmezett költségfüggvény. Egy maximális pozitív, az r pontot tartalmazó részfát szeretnénk meghatározni.

Jelölje $T(v)$ a v gyökerű részfát, $S(v)$ v gyerekeinek halmazát, $H(v)$ pedig az optimális megoldást $T(v)$-n. Ha ilyen nem létezik, $H(v)$ értéke legyen 0.

A levelekből indulva számíthatjuk ki $H(v)$ értékeit a következőképpen: ha v levél, nyilván teljesül a $H(v) = max\{c_v, 0\}$ összefüggés, ha pedig v belső pont, $H(v) = max\{0, c_v + \sum_{w \in S(v)} H(w)\}$, ugyanis ha létezik v gyökerű optimális részfa, azt úgy kaphatjuk, hogy v rákövetkező pontjainak egy-egy optimális részfájához (ami esetleg az üreshalmaz 0 költséggel) hozzávesszük a v pontot.

Ha $H(v)$ értékét már ismerjük, azon $T(v)$-ket kell törölni a fából az optimális pozitív, r-t tartalmazó részfa megtalálásához - ha van ilyen -, melyre $H(v) = 0$.

Az algoritmus $O(|V|)$ lépésszámú, mivel minden $H(v)$ meghatározása konstans sok összeadással és összehasonlítással történik.

2. fejezet

További útkeresési és részgráfokkal kapcsolatos problémák

Ebben a fejezetben bonyolultabb útkeresési és részgráfokkal kapcsolatos problémákról lesz szó.

2.1. Steiner-fa

Feladat: Adott $G = (V, E)$ irányítatlan gráf, élein egy c nemnegatív költségfüggvény, valamint csúcsainak egy T részhalmaza, melynek elemeit termináloknak nevezzük. A cél olyan T-t tartalmazó összefüggő részgráf meghatározása, amely minimális összsúlyú.

Világos, hogy ha ez a részgráf tartalmaz kört, az csak úgy képzelhető el, hogy minden él költsége nulla. Ugyanis ha lenne pozitív költségű él, azt törölve kisebb összsúlyú, ugyanolyan ponthalmazú, továbbra is összefüggő részgráfot kapnánk. Egy csupa nulla élekből álló körből viszont valamely élt törölve a kapott részgráf továbbra is megfelelő lesz. Tehát a minimum felvétetik egy részfán, és a cél ennek megtalálása. Ezt a részfát **Steiner-fának** hívják.

Ha $|T| = 2$, a feladat tulajdonképp a minimális út meghatározása a két terminálpont között, ha pedig $T = V$, akkor egy minimális költségű feszítőfa keresése.

Bár ezek polinomiális időben megoldhatók, R. Karp [8] bebizonyította, hogy közös általánosításuk, a Steiner-fa problémája NP-teljes még $c \equiv 1$ esetén is.

Dreyfus és Wagner [9] kimutatta, hogy dinamikus programozás segítségével megadható olyan algoritmus, amely $|V|$-ben polinomiális, és csak $|T|$-ben exponenciális. Tehát rögzített T-re a futásidő már polinomiális lesz. Például $|T| = 3$-ra $\forall v \in V$-re meghatározunk a terminálpontokba egy-egy minimális költségű utat, és amelyik v-re ezen utak uniója minimális, az adja az optimális Steiner-fát.

A következő jelöléseket vezessük be:

$P[uv]$ jelölje az u és v pontok közti (egyik) minimális költségű utat. Ennek költsége $c(P[uv]) \geq 0$.

$s_b(K+v)$ értékét a következőképp definiáljuk: ha a $K \cup v$ terminálhalmazhoz tartozó minimális Steiner-fák között van olyan, amelyben v belső pont, akkor $s_b(K+v)$ értéke legyen ennek a fának a költsége, ha pedig ilyen nincs, akkor az érték legyen végtelen.

$s(K+v)$ legyen a $K \cup v$ terminálhalmazhoz tartozó minimális Steiner-fa költsége.

Az algoritmus során $\forall K \subset T$-re és $\forall v \in V$-re kiszámoljuk az $s_b(K+v)$, majd az $s(K+v)$ értékeket. Ezek száma $|V|$-ben polinomiális, és csak $|T|$-től függ exponenciálisan. A keresett minimumot azon $s(K+v)$-k egyike adja, melyekre $K \cup v = T$.

Az algoritmus k-adik fázisában ($k = 1, 2, \ldots, |T| - 2$) már $\forall v \in V$-re, és T-nek $\forall$ k-elemű T' részhalmazára kiszámítottuk $s(T'+v)$-t. Ekkor ennek segítségével először meghatározzuk T $(k+1)$-elemű K részhalmazaira $s_b(K+v)$-t, majd $s(K+v)$-t az alább látható módon.

$|K| = 1$ esetén $\forall$ $s_b(K+v)$ értéke legyen végtelen, $s(K+v)$ pedig a minimális út költsége.

K legyen a terminálok egy részhalmaza, F pedig egy ezekre vonatkozó minimális Steiner-fa, melyben a v pont minimum másodfokú. A v-ből kiinduló faéleket két nemüres osztályba sorolva meghatározzuk a fának egy olyan két részre osztását, melyeknek egyetlen közös pontja v. A két részfát jelölje F', illetve F'', a hozzájuk tartozó terminálhalmazokat pedig K', illetve K''. Ezekre nyilván igaz, hogy F' mi-

nimális Steiner-fa a $K' \cup v$, F'' pedig a $K'' \cup v$ terminálhalmazra nézve. Ha nem így lenne, a minimálisnál nagyobb költségű részt kicserélve egy kisebb költségűre és a két részt újra egyesítve a K terminálhalmazra nézve egy kisebb költségű Steiner-fát kapnánk, ami ellentmond F optimalitásának.

Ebből adódik a következő rekurzió:

$$s_b(K+v) = min\{s(K'+v) + s(K-K'+v) : \oslash \subset K' \subset K\}. \tag{2.1}$$

Ha nem létezik olyan minimális Steiner-fa, amelynek v belső pontja, akkor ugyan a rekurzió hamis eredményt adhat $s_b(K+v)$ értékére, de később látni fogjuk, hogy $s(K+v)$ értékét ez nem rontja el.

$s(K+v)$ kiszámításához tekintsünk egy $K \cup v$ terminálhalmazhoz tartozó F minimális Steiner-fát. Ekkor három eset lehetséges:

1. v F-ben minimum másodfokú, ekkor $s(K+v) = s_b(K+v)$ adódik.

   $$\alpha_1 := s_b(K+v).$$

2. v az F-ben elsőfokú. Addig haladunk a fában v-től visszafelé, amíg elérünk egy olyan u pontot, amely vagy K-beli, vagy legalább harmadfokú. Ilyen biztosan van, mert ha nem lenne, az azt jelentené, hogy a fa egyetlen olyan útból áll, amely nem tartalmaz K-beli pontot.

   Aszerint, hogy u milyen, újabb két esetet különbözetünk meg.

   (a) u nem K-beli, a fában minimum harmadfokú.

   Ekkor $s(K+v) = s_b(K+u) + c(P[uv])$.

   $$\alpha_2 := min\{s_b(K+u) + c(P[uv]) : u \notin K\}.$$

   (b) u K-beli.

   Ekkor $s(K+v) = s(K) + c(P[uv])$.

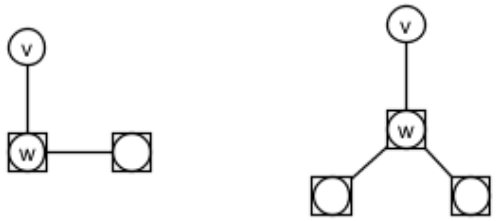
   $$\alpha_3 := min\{s(K) + c(P[uv]) : u \in K\}.$$

Ezek alapján $s(K+v)$ a következőképp számítható:

$$s(K+v) = min\{\alpha_1, \alpha_2, \alpha_3\}. \tag{2.2}$$

A fenti (2.1) és (2.2) összefüggések segítségével meghatározható a minimális Steiner-fa költsége, de maradt még egy lezáratlan kérdés: elrontja-e a végeredményt az, ha néhány $s_b(K+v)$ kiszámításakor a rekurzió végtelen helyett valamilyen más, kisebb értéket ad. Ez akkor fordulhatna elő, ha valamelyik rossz $s_b(K+v)$-vel vagy rossz $s_b(K+u)$-val számolt α_i értéke túl alacsony lenne. Tegyük fel, hogy a számolt értékek között eddig nem szerepelt hibás, és nézzük meg, mi történhet az algoritmus következő fázisában az újonnan számolt $s_b(K+v)$-k felhasználása során:

1. Tegyük fel, hogy α_2 számításakor nem lép fel hiba. Baj még akkor fordulhatna elő, ha a $K \cup v$ terminálhalmazon vett $\forall$ F minimális Steiner-fában v levél. Két esetet kell megkülönböztetni:



   (a) v-ből F-ben visszafelé haladva először olyan pontot érünk el, amely K-beli. Ekkor tévesen $s_b(K+v) = s(K) + 2 \cdot c(P[vw])$-t kapunk.
   $\alpha_1 = s_b(K+v) = s(K) + 2 \cdot c(P[vw])$.
   $\alpha_2 = min\{s_b(K+u) + c(P[uv]) : u \notin K\}$.
   $\alpha_3 = min\{s(K) + c(P[uv]) : u \in K\} = s(K) + c(P[vw])$.
   Itt a jó eredmény $\alpha 3$, és nyilván $\alpha_3 \leq \alpha_1$ teljesül $c(P[vw]) \geq 0$ miatt. Mivel feltettük, hogy α_2 nem hibás, azt nem kell vizsgálni.

   (b) v-ben visszafelé haladva először olyan pontot érünk el, amely nem K-beli, de legalább 3-fokú a fában. Ekkor pedig $s_b(K+v) = s_b(K+w)+2 \cdot c(P[vw])$ lesz az eredmény, ahol $s_b(K+w)$ értéke helyes. Ugyanis ha ez az érték hibás lenne, létezne kisebb költségű Steiner-fa a $K \cup w$ terminálhalmazon, de akkor arra kicserélve a jelenlegi fa $P[wv]$-n kívüli részét, kisebb költségű, $K \cup v$ terminálhalmazú Steiner-fához jutnánk, ami ellentmondana annak, hogy az éppen vizsgált fa optimális.

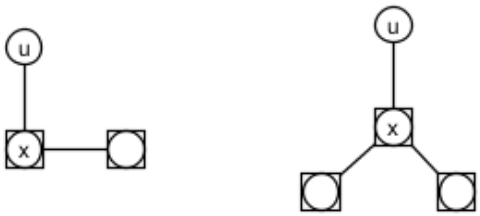
$\alpha_1 = s_b(K + v) = s_b(K + w) + 2 \cdot c(P[vw])$.

$\alpha_2 = min\{s_b(K + u) + c(P[uv]) : u \notin K\} = s(K) + c(P[vw])$.

$\alpha_3 = min\{s(K) + c(P[uv]) : u \in K\}$.

Itt a jó eredmény α_2, és nyilván teljesül $\alpha_2 \leq \alpha_1$, α_3 pedig azért nem lehet kisebb, mert feltettük, hogy a minimális Steiner-fa $K \cup v$-n a jobb oldali ábrának megfelelő alakú. Ebben az esetben pedig $c(P[vw]) \leq c(P[uv])$ $\forall u \in K$-ra.

2. Tegyük fel, hogy α_2 számításakor lép fel hiba. Ez azt jelenti, hogy α_2-re kapjuk a legkisebb értéket valamely u mellett, és ez az érték alacsonyabb minden megfelelő ponthalmazú fa összsúlyánál. Ez kétféleképp történhetne meg.



(a) $s_b(K + u)$ értékét tévesen $s(K) + 2 \cdot c(P[ux])$-ként határoztuk meg, ahol $x \in K$, és $\alpha_2 = s(K) + 2 \cdot c(P[ux]) + c(P[uv])$ lett a minimális az α_i-k között, holott $s(K + v)$ értéke valójában nagyobb.

Ekkor viszont ellentmondásra jutunk, ugyanis a K terminálhalmaz által meghatározott fához a $P[ux]$ és $P[uv]$ utak hozzávételével kapott fa értéke legfeljebb $s(K) + c(P[ux]) + c(P[uv])$. Tehát létezik α_2 értékénél nem nagyobb összsúlyú, K-t és v-t tartalmazó fa.

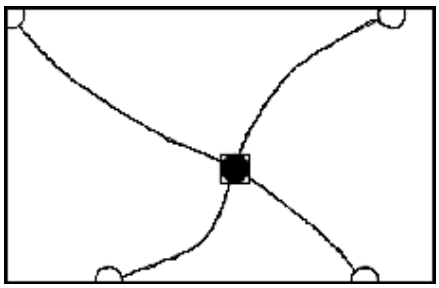
(b) $s_b(K + u)$ értékét tévesen $s_b(K + x) + 2 \cdot c(P[ux])$-ként határoztuk meg, ahol $x \notin K$, és $\alpha_2 = s_b(K + x) + 2 \cdot c(P[ux]) + c(P[uv])$ lett a minimális, holott $s(K + v)$ értéke valójában nagyobb.

Ekkor viszont szintén ellentmondásra jutunk, ugyanis annak a fának az értéke, melyet a $K \cup x$ terminálhalmaz által meghatározott fához a $P[ux]$ és $P[uv]$ utak hozzávételével kapunk legfeljebb $s(K)+c(P[ux])+c(P[uv])$.

Tehát létezik α_2 értékénél nem nagyobb összsúlyú, K-t és v-t is tartalmazó fa.

Tehát az $s_b(K+u)$-re felírt rekurzió bár téves eredményt adhat abban az esetben, ha $s_b(K+u)$ értékének végtelennek kellene lennie, ez nem befolyásolja egyik $s(K+v)$ értékét sem, vagyis a végeredményt sem.

Az algoritmus tehát jó.



Abban az érdekes esetben, ha a gráf síkbarajzolható, és a terminálpontok a végtelen tartomány határán vannak, a fenti algoritmus nem csak $|V|$-ben, de $|T|$-ben is polinomiális lesz.

Vegyünk egy tetszőleges F Steiner-fát. Ha egy élt elhagyunk F-ből, a fa két komponensre esik szét. A komponensek a terminálpontokat egymást követően tartalmazzák, tehát ha $t_1, \ldots, t_4 \in T$ ebben a sorrendben következnek a külső ablakon, nem tartozhat t_1 és t_3 az egyik, míg t_2 és t_4 a másik komponenshez. Ugyanis ha ez volna a helyzet, a fában lévő t_1t_3- és t_2t_4-utak diszjunkt komponensekhez tartoznának, holott a síkbarajzolhatóság miatt ezen utaknak van közös pontja, ami ellentmondásra vezet.

Emiatt elég az algoritmus során a terminálhalmaznak csak azon K részhalmazaira kiszámítani $s_b(K+v)$-t és $s(K+v)$-t, melyek a külső ablakon folytonosak. (Azaz $K = \{t_i, \ldots, t_j\}$) valamely $1 \leq i \leq j \leq k$-ra, ahol a $T = \{t_1, t_2, \ldots, t_k\}$ terminálpontok ebben a sorrendben következnek a külső ablakon.) Ilyen K kevesebb, mint $|T|^2$ van, így az algoritmus polinom időben megtalálja az optimumot.

2.2. st-, ill. ts-utat tartalmazó minimális részgráf

Feladat: Adott egy $G = (V, E)$ irányított gráf, illetve $s, t \in V$ pontok. Egy olyan minimális pontszámú H^* részgráfot szeretnénk meghatározni, mely tartalmaz s-ből t-be vezető, valamint t-ből s-be vezető utat is.

A következő algoritmust J. Feldman és M. Ruhl [5] konsturálta:

Definiáljuk a G' gráfot a $V \times V$ pontosztályon, az éleket pedig értelmezzük a következő módon:

1. $(u, x) \to (v, x)$ és $c((u, x), (v, x)) := 1$ $\forall uv \in E$, $x \in V \setminus \{v\}$-re.
2. $(x, v) \to (x, u)$ és $c((x, v), (x, u)) := 1$ $\forall uv \in E$, $x \in V \setminus \{u\}$-ra.
3. $(u, v) \to (u, u)$ és $c((u, v), (u, u)) := 0$ $\forall uv \in E$
4. $(u, v) \to (v, v)$ és $c((u, v), (v, v)) := 0$ $\forall uv \in E$
5. $(x, y) \to (y, x)$ és $c((x, y), (y, x)) := k - 1$ $\forall x, y \in V$, $x \neq y$, $\exists$ xy-út, ahol k a minimális xy-út hossza.

Ezen G' gráfban keressünk minimális költségű p^* utat az előző fejezetben megismert algoritmussal az (s, s) pontból a (t, t) pontba.

Állítás: Minden $p \in G'$, (s, s)-ből (t, t)-be vezető út generál egy G-beli $H(p)$ részgráfot, amely tartalmaz utat s-ből t-be és t-ből s-be.

Bizonyítás: A pontok első komponense egy utat határoz meg s-ből t-be az 1., 4. és 5. típusú élek használatával, míg a második komponens egy utat ad t-ből s-be visszafelé bejárva azt a 2., 3. és 5. típusú élek használatával.

Állítás: Minden $p \in G'$, (s, s)-ből (t, t)-be vezető útra fennáll a következő egyenlőtlenség: $wt(p) \geq |H(p)| - 1$, ahol $wt(p)$ jelenti a p út összköltségét.

Bizonyítás: Minden esetben, amikor egy új csúcsnevet vezetünk be, $wt(p)$ értékét növeljük 1-gyel, kivéve az s kezdőcsúcsot. Ezért $wt(p) + 1$ felső korlátja $|H(p)|$-nek,

mivel $H(p)$ a p-ben előforduló csúcsokat tartalmazza.

Állítás: Létezik olyan G'-beli p' út (s,s)-ből (t,t)-be, melyre $wt(p') = |\mathrm{H}*| - 1$, ahol H^* a keresett optimális részgráf. (Ez az állítás magában foglalja azt is, hogy p' optimális részgráfot határoz meg, illetve hogy G'-ben p' minimális út (s,s)-ből (t,t)-be.)

Bizonyítás: H^* legyen egy optimális részgráf, és jelölje p_{st} a H^*-beli legrövidebb st-utat, p_{ts} pedig a H^*-beli legrövidebb ts-utat. Legyen $S = \{v_1, \ldots, v_k\}$ azon s-től különböző pontok halmaza, melyeket p_{st} és p_{ts} is tartalmazza.

Konstruálunk egy p'-t oly módon, hogy mindig megnöveljük 1-gyel a költségét abban az esetben, ha új pontot érünk el. Induljunk ki az (s,s) pontból, és amíg S-beli pontot nem érünk el, 1. típusú éleket használva, tehát az első komponensben haladva építsük p'-t p_{st} mentén haladva. Legyen az így elért S-beli csúcs u. Hasonlóképpen p_{ts} mentén s felől visszafelé haladva 2. típusú éleket használva építsük p'-t, míg el nem érünk egy S-beli pontot, amit nevezzünk v-nek. Ennek során mindig 1 költséggel veszünk hozzá új éleket p'-höz, és mindig egy új pontot érünk el, kivéve esetleg a legutolsó lépésben, ha $u = v$, mert akkor egy 3. típusú, 0 költségű élt használtunk utoljára, de ekkor nem is vettünk be új pontot. Tehát a számolás eddig helyes.

Ha $u \neq v$, akkor abból az következik, hogy v valamikor u után fordul elő a p_{st} út során, u pedig valamikor v előtt kerül sorra a p_{ts} úton, mivel $u, v \in S$.

Mivel p_{st}, illetve p_{ts} minimálisak voltak, ezek uv-részútjának - mely nyilván ugyanolyan ponthalmazú mindkettőben, hiszen elég egyféleképp eljutni u-ból v-be, és H^* minimális volta miatt fölösleges csúcsokat nem használunk - is minimálisnak kell lennie, különben a megfelelő részt kicserélve az eredeti utak helyett olcsóbbakat kapnánk.

Az (u,v) pontból 5. típusú élt használva eljuthatunk a (v,u) pontba, és mivel a H^*-beli költség k volt, a H^*-beli uv-útnak k éle, következésképp $k-1$ belső pontja van, vagyis ennek az éltípusnak a használatával is pontosan annyival növeltük a p' út költségét, mint ahány új pontot elértünk.

Folytassuk a fentieket, míg el nem jutunk (t,t)-be. A fenti elemzés végig érvényes marad, hiszen minden lehetséges esetet végignéztünk, így a G'-beli p' út költsége a

H^*-beli pontok költségénél 1-gyel kevesebb, hiszen s bevezetésekor a költség még 0 volt, és utána minden újonnan elért csúcsnál nőtt pontosan 1-gyel. Tehát a bizonyítandó igaz.

Sikerült visszavezetni a G-beli st-, ill. ts-utat tartalmazó minimális részgráf megkeresését G'-beli (s, s)-ből (t, t)-be vezető minimális út keresésének problémájára, amit már meg tudunk oldani az előző fejezetben megismert módon.

A feladat könnyen megoldható pontsúlyozott esetben is, ekkor az 1. és 2. típusú élek költségeit definiáljuk az újonnan bevezetett pont súlyának megfelelően, az 5. típusú él esetében pedig szintén az út során felmerülő $k-1$ db belső pont súlyának összege adja az élköltséget. Erre az esetre a bizonyítás teljesen ugyanúgy zajlik, mint súlyozatlan esetben.

A feladatot úgy is módosíthatjuk, hogy adott v_1, v_2, u_1, u_2 pontokra szeretnénk egy olyan minimális részgráfot, melyben van u_1v_1-, illetve u_2v_2-út. Ekkor G-t módosítsuk a következőképpen: vegyünk fel két új pontot, s-t és t-t, és négy új élt: s-ből u_1-be, v_1-ből t-be, t-ből u_2-be és v_2-ből s-be. Az eredeti probléma megfelel ebben az új gráfban egy st-, ill. ts-utat tartalmazó minimális részgráf keresésének.

Ha pedig olyan értelemben szeretnénk módosítani a feladatot, hogy H^* minimális élszámú legyen, ezt megtehetjük úgy, hogy minden élt kettéosztunk egy ponttal, és ebben az új gráfban keresünk minimális részgráfot. Erre azért van szükség, mert eddig előfordulhatott, hogy olyan értelemben fölösleges éleket is belevettünk H^*-ba, amelyek a pontok számát nem növelték, de az élek számát igen.

2.3. Pont-, illetve éldiszjunkt s_it_i-utak

Feladat: Adott egy aciklikus $G = (V, E)$ digráf, illetve $s_1, \ldots, s_k, t_1, \ldots, t_k \in V$ csúcsok. A feladat annak eldöntése, hogy létezik-e pontdiszjunkt $s_1t_1, \ldots, s_kt_k$-út.

Ez a probléma a pontpárok számát, vagyis k-t paraméternek tekintve NP-teljes, visznt rögzített k-ra hatékonyan megoldható.

Mivel a gráf aciklikus, kereshetünk egy topológikus sorrendet, és ebben a sorrendben vizsgálhatjuk a pontokat.

Definiáljunk egy $f(x_1, \ldots, x_k)$ függvényt, amelynek értéke igen, ha létezik pontdiszjunkt $s_i x_i$-út $\forall i \in \{1, \ldots, k\}$-ra.

$f(x_1, \ldots, x_k) = \text{igen} \Leftrightarrow \exists u_1, \ldots, u_k \in V$: $u_1 x_1, \ldots, u_k x_k \in E$, $f(u_1, \ldots, u_k) = igen$, és az x_i-kra, valamint az $f(u_1, \ldots, u_k)$ során használt legalább az egyik út pontjaira igaz az, hogy páronként nemegyenlők. Tehát az $f(x_1, \ldots, x_k)$ igazságértékei mellett nyilván kell tartani az odajutáshoz használható lehetséges utak pontjait. (Ez úgy történhet, hogy az $f(x_1, \ldots, x_k)$ értékének eldöntésekor használt és helyesnek talált $f(u_1, \ldots, u_k)$-kkal együtt tárolt utak pontjaihoz hozzávesszük az x_i-ket.)

Ez $\binom{n}{k}$ pont k-ast jelent, rögzített k-ra a részproblémák száma polinomiális, rögzítetlen k-ra viszont nem.

Minden $f(x_1, \ldots, x_k)$ számításakor a belépő él k-asokat kell megvizsgálni.

Ha pontdiszjunktság helyett éldiszjunktságot írunk elő, hasonló függvényt definiálhatunk, csak akkor nem a pontok, hanem a használt élek páronkénti különbözőségét kell ellenőrizni.

3. fejezet

Részbenrendezett halmazzal kapcsolatos problémák

Ebben a fejezetben a részbenrendezett halmazban keresett maximális lánc problémájáról, illetve erre visszavezethető egyszerűbb feladatokról lesz szó.

3.1. Maximális súlyú lánc

Feladat: Adott egy P n-elemű részbenrendezett halmaz, az elemeken pedig egy c súlyozás. Keresünk egy maximális súlyú láncot, vagyis maximális súlyú teljesen rendezett részhalmazt.

Ha C maximális lánc és $p \in C$, akkor $\forall q > p$ -t elhagyva olyan láncot kapunk, amely maximális súlyú a p maximális elemű, vagyis p-nél nagyobb elemet nem, de p-t tartalmazó láncok között. Ha lenne egy nagyobb összsúlyú p maximális elemű lánc, akkor C p-nél kisebb elemeit elhagyva, és ezt a láncot írva a helyére továbbra is teljesen rendezett részhalmazt kapnánk, viszont nagyobb súllyal, ami ellentmond C optimalitásának.

Ez azt jelenti, hogy a feladat egy optimális megoldása hasonló alfeladatok optimális megoldásából épül fel.

Az optimumot megpróbáljuk tehát alulról felfelé építkezve megtalálni.

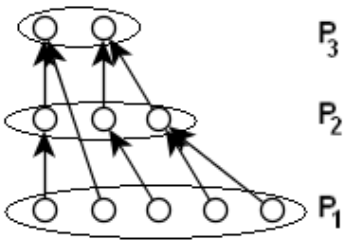
$m(p)$ legyen a p maximális elemű láncok maximális súlya.

P_i legyen P i. legkisebb elemeinek halmaza, azaz P_1 jelölje a P-beli legkisebb elemeket, P_2 a $P-P_1$-beli legkisebb elemeket,..., P_{i+1} pedig a $P-P_1-P_2-\cdots-P_i$-beli legkisebb elemeket.

P_1 elemeire nyilván teljesül az $m(p) = c(p)$ egyenlőség, mivel P_1-ben csupa egyelemű lánc van. Tegyük fel, hogy $m(p)$ értéke ismert $P_1 \cup \cdots \cup P_i$-n. Ekkor $p \in P_{i+1}$-re a következőképp kaphatjuk meg $m(p)$-t annak a fenti összefüggésnek a felhasználásával, hogy egy p_n maximális elemű maximális láncot egy olyan p_m maximális elemű maximális láncból kapjuk p_n hozzávételével, melyre teljesül a $p_n > p_m$ összefüggés.:

$m(p) = c(p)+max\{m(u)\}$, ahol $u \in P_j \quad (1 \leq j \leq i)$, és $u < p$ a részbenrendezésben.

Ilyen módon miután P-t felosztottuk a fent látható módon P_i halmazokra, az algoritmus k-adik fázisa során kiszámoljuk $m(p)$-t $\forall p \in P_k$-ra. A maximális lánc súlyát az $m(p)$-k maximuma adja.



Megjegyzés: Ha elkészítjük P fent szereplő P_i csoportokra osztását, a relációban lévő elemeket pedig irányított élekkel kötjük össze, ahol egy él kezdőpontja a részbenrendezésben kisebb, mint a végpontja, akkor egy aciklikus, irányított, pontsúlyozott G gráfot kapunk. Egy lánc a részbenrendezett halmazban egy útnak felel meg ebben a gráfban, amely a láncbeli rendezés sorrendjében tartalmazza a pontokat. Tehát egy maximális lánc keresése P-ben megfeleltethető egy maximális út keresésének az aciklikus G digráfban, amit akár átírhatunk élsúlyozott aciklikus digráfban minimális út keresésére is, ha negáljuk a költségfüggvényt, és minden pontot megkettőzünk, a köztük vezető élekre a pont költségét írjuk, a többi élre pedig 0-t.

A következő alfejezetekben olyan problémákról lesz szó, melyek nagyon hasonló struktúrájúak, és visszavezethetők részbenrendezett halmazban maximális lánc keresésére.

3.2. Maximális monoton növő részsorozat

Feladat: Adottak az $a_1, a_2, \ldots, a_n$ különböző számok. A feladat maximális monoton növő részsorozat meghatározása.

Itt is érvényes egy hasonló összefüggés, mint amit az előbb láthattunk:

Ha M maximális monoton növő részsorozat és $a_i \in M$, akkor az $a_1, \ldots, a_i$ számok között az a_i-t tartalmazó monoton növő részsorozatok közt maximális lesz M a_i-ig terjedő része.

Ezen megfigyelés alapján hasonlóképp megoldható a probléma, mint az előző esetben, de megfelelő részbenrendezést és súlyozást definiálva közvetlenül is visszavezethető rá.

Legyen $P = \{p_1, ..., p_n\}$ részbenrendezett halmaz, p_i jelöli a_i-t, a részbenrendezés pedig a következő: $p_i < p_j$, ha $i < j$ és $a_i < a_j$. A költségfüggvény legyen azonosan 1.

A reláció jelenti a monoton növekedő tulajdonságot, a költségfüggvény pedig azért lesz 1, mert csak a hosszúságra vagyunk kíváncsiak, nincsenek megkülönböztetett elemek. Ha bizonyos elemeket mindenképp be akarnánk választani a sorozatba, más súlyozást is lehetne definiálni.

Ez a rendezés jóldefiniált, mivel ha $p_i < p_j < p_k$, akkor $i < j < k$ és $a_i < a_j < a_k$, tehát $i < k$ és $a_i < a_k$, így $p_i < p_k$ is teljesül.

3.3. Maximális keresztezésmentes párosítás

Feladat: Adott $G = (A, B; E)$ páros gráf, $A = a_1, \ldots, a_k$, $B = b_1, \ldots, b_l$. Keresünk M maximális keresztezésmentes párosítást, tehát ha $a_i b_j$, $a_{i'} b_{j'} \in M$, és $i < i'$, akkor $j < j'$.

Ha egy M-beli él mentén kettévágjuk a gráfot, az általa meghatározott két részen külön-külön is maximális keresztezésmentes párosításokat kapunk. Tehát felépíthető a rekurzió egy választott élig keresve maximális keresztezésmentes párosítást. Ez megint az előzőhöz hasonló optimalitási tulajdonság, ismét vissza lehet vezetni a problémát maximális lánc keresésére.

A részbenrendezést az éleken értelmezzük a következőképp: $a_ib_j < a_{i'}b_{j'}$, ha $i < i'$ és $j < j'$. A költségfüggvény most is $\equiv 1$, ha az éleken nincs súlyozás, ha pedig van, akkor c-t ennek megfelelően értelmezzük.

A reláció jelöli a keresztezésmentességet. A tranzitivitás is teljesül, ugyanis ha $a_ib_j < a_kb_l < a_mb_n$, akkor $i < k < m$ és $j < l < n$, tehát $i < m$ és $j < n$, így $a_ib_j < a_mb_n$.

3.4. Maximális közös részsorozat

Definíció: $X = \{x_1, \ldots, x_n\}$ nem feltétlenül különböző betűk sorozatának $Z = \{z_1, \ldots, z_l\}$, $k \leq n$ sorozat részsorozata, ha $\exists i_1 < \cdots < i_l : x_{i_1} = z_1, \ldots, x_{i_l} = z_l$.

Feladat: Adottak $X = x_1, \ldots, x_n$ és $Y = y_1, \ldots, y_m$ sorozatok, keressük ezek maximális közös részsorozatát.

Ha $Z = \{z_1, \ldots, z_l\}$ egy maximális közös részsorozat, $z_k = x_{i_k} = y_{j_k} \forall 1 \leq k \leq l$, akkor $X' = \{x_1, \ldots, x_{i_k}\}$ és $Y' = \{y_1, \ldots, y_{j_k}\}$ sorozatok z_k-t tartalmazó közös részsorozatai között $Z' = \{z_1, \ldots, z_k\}$ maximális.

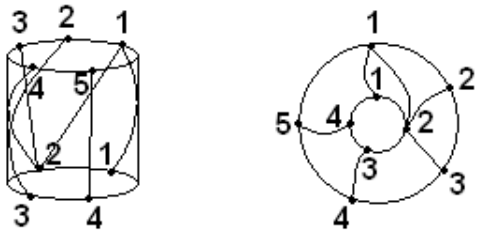
Tehát ismét felbukkant a maximális láncra jellemző tulajdonság. Mivel itt két csoport kapcsolatát vizsgáljuk, kézenfekvőbb párosgráfos alakba átírni a feladatot, mint közvetlenül részbenrendezett halmazra.

$A := \{a_1, \ldots, a_n\}$, $B := \{b_1, \ldots, b_m\}$, legyen $a_ib_j \in E$, ha $x_i = y_j$. A maximális közös részhalmaz megkeresésének feladata megfelel a $G := (A, B; E)$ páros gráfban maximális keresztezésmentes párosítás keresésének.

A keresztezésmentesség a "részsorozat", az élek pedig a "közös" tulajdonságot jelölik az X és Y sorozatok elemei közt.

3.5. Hengeren maximális keresztezésmentes élhalmaz keresése

Feladat: A feladat nagyon hasonlít a páros gráfokban keresett maximális keresztezésmentes párosítására, de itt a két pontosztály egy henger alsó, illetve felső körlapjának peremén helyezkedik el, élek pedig csak két különböző körlapra eső pontok között haladhatnak. A cél maximális keresztezésmentes élhalmaz kiválasztása.



Világos, hogy ha egy él mentén "kettévágjuk" a hengert, azaz egy élt fixen beválasztunk a keresett élhalmazba, ezzel visszavezetjük a feladatot páros gráfban keresett maximális keresztezésmentes párosításra, mert ilyen módon megadjuk a pontok sorrendjét. Ha minden élre elvégezzük ezt, a végén kiválaszthatjuk a kapott élhalmazok közül a maximálisat.

Ha kevés él van, ez megfelelő, de ha az élek száma nagyon magas, érdemes lehet más megoldást keresni.

Számozzuk be a pontokat mindkét körlapon, az alsón 1-től n-ig, a felsőn 1-től m-ig. Tegyük fel, hogy $n \leq m$. Ekkor készítünk n darab párosgráfot a következőképpen: az A pontosztályt adják a felső pontok sorszám szerint növekvő sorrendben elhelyezve, a B pontosztály pedig álljon az alsó pontok egy sorrendjéből, ami az i-edik gráf esetében legyen a következő: $i, i+1, \ldots, n-1, n, 1, 2, \ldots, i-1$. Ez azért lesz megfelelő, mert ha M egy optimális élhalmaz, a felső pontok közül M az i-ediket tartalmazza végpontként, a nála kisebb sorszámúakat pedig nem, és ennek a pontnak M-ben az alsó ponthalmaz j-edik eleme a párja, akkor a j-edik párosgráfban ennek az élnek az alsó végpontja a B-ben első helyen fog szerepelni, tehát tulajdonképpen ugyanazt csináltuk, mint az élelvágások esetén, csak a pontsorren-

deken keresztül végezve el. Tehát mindegyik párosgráfban megkeressük a maximális keresztezésmentes párosítást, és ezek közül kiválasztva az optimálisat, megkapjuk a feladat egy megoldását.

4. fejezet

Különböző problémák közötti kapcsolatok

Ebben a fejezetben olyan különböző problémákról lesz szó, melyek visszavezethetőségi szempontból érdekesek.

4.1. Maximális konvex részhalmaz

Feladat: Adott a síkon n darab pont, tegyük fel, hogy ezek első koordinátái páronként különbözők, és nincs köztük három egy egyenesre eső. A feladat az, hogy keressük meg ezen pontok egy maximális konvex részhalmazát.

Először meg kell határozni a feladatnak egy olyan alfeladatokkal bővítését, amely a dinamikus programozás feltételeinek megfelel.

Állítás: Adott x-re és y-ra minden őket tartalmazó olyan maximális konvex részhalmaz (K), melyben minden pont első koordinátája x és y első koordinátái közé esik, előáll úgy, mint az xy-szakasz fölötti (K_1), illetve az xy-szakasz alatti (K_2), x-t és y-t tartalmazó maximális konvex részhalmazok egyesítése, ahol egy pontot akkor nevezünk az xy-szakasz fölöttinek/alattinak, ha első koordinátája x és y első koordinátái közé esik, a második koordináta pedig értelemszerűen viszonyul

az adott szakaszhoz.

Jelölje $d_1(x, y)$ azon pontok halmazát, melyek az xy-szakasz fölé, $d_2(x, y)$ pedig azon pontok halmazát, melyek az xy-szakasz alá esnek.

Legyen $f_1(x, y)$ az xy-szakasz fölötti pontok közül, $f_2(x, y)$ pedig az xy-szakasz alatti pontok közül kiválasztható maximális konvex részhalmaz.

Az $f_1(x, y)$, ill. $f_2(x, y)$ meghatározásához szükség lesz segédfüggvényekre: $f_1(x, y; z)$ jelölje azon xy-szakasz fölötti maximális konvex részhalmazt, mely szétosztható egy xz-szakasz és egy zy-szakasz fölötti (z-t nem feltétlenül tartalmazó) konvex részhalmazra. $f_2(x, y; z)$-t hasonlóan definiáljuk.

$|d_1(x, y)| = 0$ esetén $f_1(x, y) = \{x, y\}$.

Tegyük fel, hogy $|d_1(x, y)| > 0$. Ekkor $f_1(x, y; z)$ kétféle lehet attól függően, hogy z eleme lesz a halmaznak, vagy sem:

$f_1(x, y; z) = f_1(x, z) \cup f_1(z, y) \setminus \{z\}$, ha z a w_1w_2 szakasz alatt van, ahol w_1 az $f_1(x, z)$ z előtti utolsó pontja, w_2 pedig az $f_1(z, y)$ z utáni első pontja;

$f_1(x, y; z) = f_1(x, z) \cup f_1(z, y)$, ha z a w_1w_2 szakasz felett van.

$f_1(x, y) \in \{f_1(x, y; z) : max_w|f1(x, y; w)| = |f1(x, y; z)|\}$.

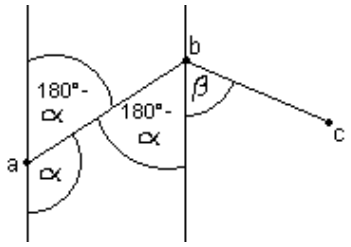
Ha már minden pontpárra kiszámítottuk az $f_1(x, y)$-t, a ponthalmazt tükrözve kiszámoljuk ismét, vagy az $f_2(x, y)$ függvényt számoljuk ki hasonlóképpen a szakaszok alatti részekre, hogy megkapjuk a keresett konvex részhalmaz alsó részét. Minden x,y pontpárra egyesítjük a két eredményt, és kiválasztjuk a maximálisat.

Visszavezetés aciklikus digráfban maximális út keresésére :

A pontpárokat megfeleltetjük egy G gráf csúcsainak, az élhalmazt pedig a következőképpen definiáljuk:

$(u_1, u_2) \to (v_1, v_2)$, ha $u_2 = v_1$ és u_2 az u_1v_2 szakasz fölött van.

Állítás: Az így definiált gráfban egy maximális xy-út megfelel egy xy-szakasz fölötti, x-t és y-t tartalmazó maximális konvex ponthalmaznak.



Bizonyítás: Egy él egy olyan ponthármast ír le, amelynek középső pontja a szélsők által meghatározott szakasz fölé esik. Tekintsük az $(a, b) \to (b, c)$ élt. Az ab-szakasz függőlegessel bezárt szöge legyen α, a bc-szakasz függőlegessel bezárt szöge pedig legyen β. Ha $\alpha = \beta$, az jelenti azt, hogy a három pont egy egyenesre esik, ha $\alpha < \beta$, akkor a középső pont a szélsők által meghatározott szakasz alá, ha pedig $\alpha > \beta$, akkor a szakasz fölé esik. Ez viszont azt jelenti, hogy a három pont által meghatározott abc-szög $\beta + 180° - \alpha < 180°$ (mivel β-hoz α kiegészítő szögének váltószöge adódik hozzá). Ezt azt jelenti, hogy a kapott sokszögvonal minden szöge hegyesszög.

A gráf aciklikus, mert ha lenne benne kör, az azt jelentené, hogy eljutottunk x-ből x-be egy xx-szakasz fölötti konvex sokszögvonallal.

Tehát a gráfban kereshetünk minden pontpárra minimális utat, ha az élek költségét $\equiv -1$-nek választjuk, s ennek segítségével minden pontpárra meghatározhatunk egy fölöttük lévő, őket tartalmazó maximális konvex részhalmazt, majd ugyanezt hasonlóan megtehetjük az alattuk lévő maximális konvex részhalmazzal, s a két eredmény összegének kereshetjük a maximumát, ez adja a keresett maximális konvex részhalmazt.

4.2. Súlyozott intervallum probléma

Feladat: Adott egy I intervallumrendszer, valamint az intervallumokon értelmezett c költségfüggvény. A cél egy maximális diszjunkt részrendszer meghatározása. (A végpontok egyezését most megengedjük, de teljesen hasonlóan megoldható a feladat akkor is, ha nem.)

Az osztópontok sorrendben a következők: $v_0, \ldots, v_n$. Jelölje $f(v_k)$ a $[v_0, v_k]$ inter-

vallumon vett maximális diszjunkt részrendszert. $f(v_0) = 0$, és ha már tudjuk $f(v_i)$ értékét $\forall i < k$-ra, $f(v_k)$-ra a következő rekurziót írhatjuk föl:

$$f(v_k) = max\{f(v_{k-1}), \max_{[v_i,v_k]\in I}\{c([v_i, v_k]) + f(v_i)\}\}$$

Ugyanis már számításba kell venni a $v[k]$ végpontú intervallumokat is, és vagy nem választunk ki ezek közül egyet sem, ekkor $f(v_k) = f(v_{k-1})$, vagy beválasztunk a részintervallum-rendszerbe egy $[v_i, v_k]$ intervallumot, és ehhez még hozzáveszünk egy $[v_0, v_i]$-n optimális részrendszert.

A súlyozott intervallum probléma visszavezetése részbenrendezett halmazban maximális lánc keresésére:

Definiáljunk egy részbenrendezést a következőképpen: jelentse a_i az i. intervallumot, és legyen $a_i < a_j$, ha a_i végpontja $\leq a_j$ kezdőpontjánál. Az a_i elem súlya egyezzen meg az i. intervallum súlyával. Ez valóban helyes részbenrendezés, és a reláció a diszjunkt tulajdonságot, pontosabban az intervallumok diszjunkt egymásutániságát fejezi ki. Ebben a részbenrendezett halmazban a maximális lánc keresése ekvivalens probléma az optimális részintervallum-rendszer meghatározásával.

4.3. Bináris hátizsák feladat

Feladat: Adott n darab tárgy, c a tárgyakon értelmezett nemnegatív értékfüggvény, a szintén a tárgyakon értelmezett nemnegatív súlyfüggvény, és egy $b > 0$ kapacitású hátizsák. A kérdés, hogy hogyan lehet ebbe a hátizsákba a lehető legnagyobb összértékben tárgyakat elhelyezni a kapacitás figyelembevétele mellett, azaz a következő értéket szeretnénk meghatározni: $max\{\sum_{i=1}^{n} c_i x_i : \sum_{i=1}^{n} a_i x_i \leq b, x \in \{0,1\}^n\}$.

A feladatot bővítsük olyan módon alfeladatokkal, hogy a tárgyak közül csak az első k darab használatát engedélyezzük, a hátizsák kapacitását pedig d-nek választjuk. Ennek a részfeladatnak az optimumát jelöljük $f_k(d)$-vel: $f_k(d) := max\{\sum_{i=1}^{k} c_i x_i : \sum_{i=1}^{k} a_i x_i \leq d, x \in \{0,1\}^k\}$, $k \in \{1, \ldots, n\}$, $d \in \{0, \ldots, b\}$.

A feladat optimumát $f_n(b)$ adja.

$f_0(d) = 0$, a további $f_k(d)$-k értékét pedig az alábbi módon számíthatjuk ki: $f_k(d) = f_{k-1}(d)$, ha $a_k > d$, mivel ekkor a k-adik tárgyat továbbra sem tudjuk felhasználni, illetve $f_k(d) = max\{f_{k-1}(d), f_{k-1}(d - a_k) + c_k\}$, ha $a_k \leq d$, mivel ekkor vagy továbbra sem használjuk a k-adik tárgyat, vagy felhasználjuk, de akkor a többi tárgy számára fennmaradó hely csökken a k-adik súlyával.

Az algoritmus $O(nb)$ lépésszámú, mivel egy adott f_k kiszámításához konstans sok összeadás és összehasonlítás szükséges.

Egészértékû hátizsák feladat:

A fenti feladatot módosíthatjuk úgy, hogy egy tárgyat korlátlan darabszámban felhasználhatunk, azaz most $x \in \mathbf{N}^n$.

Ekkor a feladatot meg lehetne önállóan is oldani (lásd: [6]), de most érdekesebb, hogy visszavezethető súlyozott intervallum problémára is.

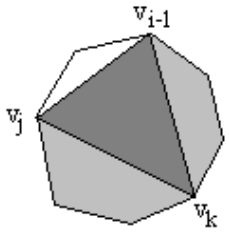
Az intervallumrendszerbe $[0, b]$ intervallumba eső intervallumok kerülhetnek. Az i-edik tárgynak feleljen meg egy a_i hosszúságú, c_i súlyú intervallum, amit minden lehetséges végpontból felmérünk. Tehát először a $v_0 = 0$-ból mérjünk fel az összes tárgynak megfelelő intervallumokat, és vegyük hozzá a végpontok halmazához az újonnan keletkezett v_j-ket. A következő lépésben mérjünk fel v_1-ből minden lehetséges intervallumot, stb, egészen addig, míg már nem mérhető fel több $[0, b]$-be eső intervallum. Egy maximális diszjunkt részintervallum-rendszer keresése megfelel az egészértékű hátizsákfeladat megoldásának, ugyanis a $[0, b]$-beliség és a diszjunktság jelenti a kapacitás figyelembe vételét, a súlyozás és a részrendszer optimális volta pedig az optimális tárgyak kiválasztását.

4.4. Minimális háromszögekre bontás

Feladat: Adott egy $P = (v_0, \ldots, v_n)$ poligon, és egy w súlyfüggvény az oldalak és átlók által meghatározott háromszögeken. Ennek a poligonnak egy minimális összsúlyú háromszögekre bontását, vagyis triangulizációját keressük.

Jelölje $t(i,j)$ a $v_{i-1},\ldots,v_j$ poligon optimális háromszögekre bontásának súlyát, $i=j$ elfajult esetben $t(i,j):=0$. $t(i,j)$-re a következő rekurziót írhatjuk fel:

$$t(i,j)=\min_{i-1<k<j}\{t(i,k)+t(k,j)+w(v_{i-1},v_k,v_j)\}$$



Ez azt fejezi ki, hogy ha beválasztjuk a v_{i-1}, v_k, v_j pontok alkotta háromszöget az optimális részháromszögelésbe, akkor ez kettéosztja a feladatot a maradék két rész optimumának, vagyis $t(i,k)$-nak és $t(k+1,j)$-nek a kiszámítására.

Az algoritmus futásideje $\Theta(n^3)$.

4.5. Mátrixok szorzássorrendje

Feladat: Adott n darab mátrix a következő sorrendben: $A_1,\ldots,A_n$, és ezeket szeretnénk összeszorozni. Feltehető, hogy a mátrixok méretei megfelelőek, tehát értelmes az összeszorzásukról beszélni ebben a sorrendben. A feladat annak meghatározása, hogy milyen zárójelezés mellett tudjuk a legkevesebb elemi szorzással kiszámítani a fenti mátrixok szorzatát, ha az A_i mátrix $p_{i-1}\times p_i$ méretű.

Egy A $p\times q$ méretű és egy B $q\times r$ méretű mátrix összeszorzásának költsége $p*q*r$, a szorzatuk mérete pedig $p\times r$-es lesz.

Jelöljük $A_{i,j}$-vel az $A_i,\ldots,A_j$ mátrixok összeszorzásának eredményét.

Egy optimális zárójelezés a következőképp áll elő: $A_1*\cdots*A_n=A_{1,k}*A_{k+1,n}$ valamely $1\le k<n$-re, mivel valahol kettévágtuk a szorzatot egy zárójellel. Ez azt jelenti, hogy az egész szorzássorozat költsége az $A_{1,k}$ és az $A_{k+1,n}$ egy optimális

kiszámítási költségéből kapható az $A_{1,k} * A_{k+1,n}$ szorzat kiszámítási költségének hozzávételével. Innen már könnyen látható, hogy egy megfelelő alfeladatokkal bővítést az $A_{i,j}$-k kiszámítása jelenthet, mégpedig $(j - i)$ nagysága szerint növekvő sorrendben.

Ehhez definiáljunk egy $m(i, j)$ függvényt, amely jelentse $A_{i,j}$ kiszámításának minimális költségét. Ha $i = j$, akkor egyetlen mátrix szorzásköltségét kell kiszámítani, vagyis $m(i, i) = 0 \ \forall i \in \{1, \ldots, n\}$ esetén. Ha $i < j$, akkor a következő rekurzióval számítható $m(i, j)$ értéke: $m(i, j) = min_{i \leq k < j}\{m(i, k) + m(k + 1, j) + p_{i-1} * p_k * p_j\}$, vagyis a már említett módon valahol kettévágjuk a szorzatot, és külön-külön kiszámoljuk a két részlet optimális megoldását, illetve egyesítésük költségét.

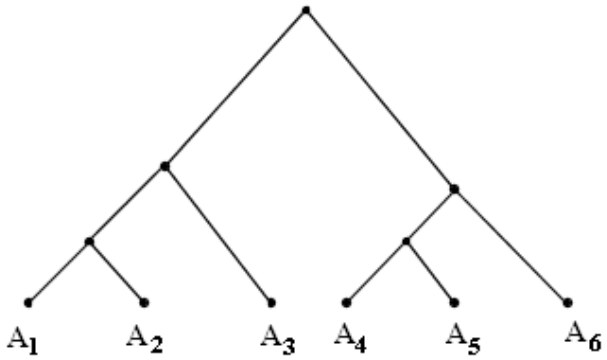
Ha a feladatot úgy próbáltuk volna megoldani, hogy minden lehetséges zárójelezés költségét meghatározzuk, és kiválasztjuk az optimálisat, exponenciális futásidőt kaptunk volna. Ennél lényegesen jobb a fenti algoritmus, mivel ennek futásideje csupán $O(n^3)$. (Pszeudokódokért lásd: [2], [3].)

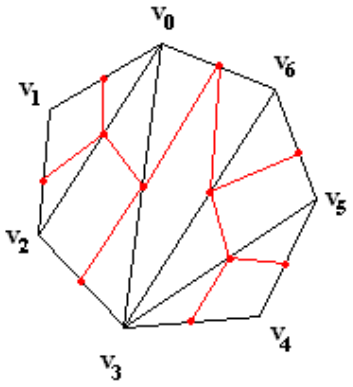
Visszavezetés minimális háromszögelésre

Egy kifejezés teljes zárójelezése megfeleltethető egy bináris fának, amit elemzési fának szoktak nevezni. A fa levelei jelölik a kifejezés egy-egy elemét, esetünkben egy-egy mátrixot. A belső pontok mutatják a zárójelezést, egy belső pont a bal és jobb oldali részfák zárójelbe foglalását jelenti. Tehát kölcsönösen egyértelműen megfeleltethetők az n-levelű elemzési fák, illetve az n-tagú kifejezések teljes zárójelezései.

Egy $n + 1$-csúcsú poligon háromszögelése szintén felírható elemzési fák segítségével. Kijelölünk egy oldalt, például a v_0v_n oldalt a fa gyökerének, a többi oldal a levelekbe kerül. A fa belső pontjai - a gyökér kivételével - az átlókat határozzák meg, mégpedig úgy, hogy egy belső pont a bal és jobb gyerekeibe tartozó oldalakat választja le a poligon többi részétől. Tehát egy rögzített gyökérpont mellett kölcsönösen egyértelmű megfeleltetés van az n-levelű elemzési fák, illetve az $n + 1$-csúcsú poligonok között.

A mátrixok optimális összeszorzásának problémája speciális esete az optimális háromszögelés feladatának.





$(((A_1A_2)A_3)((A_4A_5)A_6))$

Legyen $P = v_0, \ldots, v_n$ a kérdéses poligon, a súlyfüggvényt pedig definiáljuk a következőképpen: $w(v_i, v_j, v_k) = p_i * p_j * p_k$. Ennek a poligonnak az optimális háromszögekre bontása megadja a mátrixok egy teljes zárójelezését.

4.6. Karaktersorozatok hasonlósága

Ennek a feladatnak komoly biológiai alkalmazásai vannak, például DNS-láncok hasonlóságának vizsgálatával lehet következtetni az evolúciós folyamatokra, egyes fajok helyére az evolúciós láncban.

Feladat: Adott két karaktersorozat: $X = (x_1, \ldots, x_n)$ és $Y = (y_1, \ldots, y_m)$, és ezek különbözőségének mértékét szeretnénk úgy megállapítani, hogy adott minden karakterre, illetve karakterpárra a beszúrás (δ) és csere ($\alpha(x_i, y_j)$) műveletek költsége. Tehát a két karaktersorozatot ezen műveletek segítségével alakítjuk át úgy, hogy az eljárás végén megegyezzenek, és szeretnénk minimalizálni az egymásba alakítás költségét.

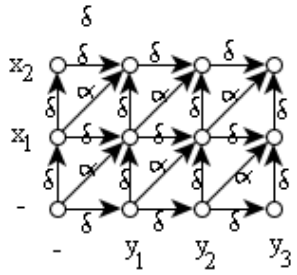
Egy S karaktersorozat első i karakterét jelölje $S(i)$, opt(i,j) pedig legyen $X(i)$ és $Y(j)$ optimális egymásba alakításának költsége. $opt(0, 0) := 0$.

$opt(i, j)$ értékét vagy úgy kapjuk meg a korábbi átalakítási optimumok ismeretében

ben, hogy x_i-t és y_j-t cserével alakítjuk egyezővé, és ennek költségéhez hozzáadjuk $opt(i-1, j-1)$ értékét - ez esetleg 0 is lehet, ha $x_i = y_j$ -, vagy utoljára egy beszúrásműveletet alkalmazunk $X(i)$ és $Y(j)$ egymásba alakításához, ekkor $opt(i-1, j)$-hez vagy $opt(i, j-1)$-hez hozzáadódik egy karakter beszúrásának költsége.

$opt(i,j) = min\{\alpha(x_i, y_j) + opt(i-1, j-1), \delta + opt(i-1, j), \delta + opt(i, j-1)\}$, $i \leq 1$, $j \leq 1$.

A futásidő $O(mn)$, mivel egy adott $opt(i, j)$ számításához csak konstans mennyiségű összeadás és összehasonlítás szükséges.



A megoldásokat fölépíthetjük visszafelé haladva is, ekkor legyen $g(n, m) = 0$, $i < n$, $j < m$ esetén pedig az alábbi rekurziót használjuk: $g(i,j) = min[\alpha(x(i+1), y(j+1)) + g(i+1, j+1), g(i, j+1) + \delta, g(i+1, j) + \delta]$. A $g(i, j)$ függvény tulajdonképpen az $X = (x_i, \ldots, x_n)$ és $Y = (y_j, \ldots, y_m)$ karaktersorozatokra oldja meg az egyezőségvizsgálatot.

Ennek a problémának az érdekességét még az adja, hogy a két függvény kapcsolatát felhasználva a kézenfekvő n-szer m-es tömböt használó kitöltés helyett más, kisebb tárhelyigényű módon is megoldható a feladat.

Az első észrevétel, hogy $opt+g$-ben az (i, j)-edik elem azt adja meg, hogy mennyi a megoldás költsége akkor, ha kikötjük (i, j)-edik elemet használatát, más szóval az optimális döntéssorozat során valamikor használnunk kell $opt(i, j)$-t $opt(n, m)$ felépítéséhez.

A második észrevétel, hogy ha $opt + g$-ben tetszőleges k mellett q az az index, ami minimalizálja $opt(q, k) + g(q, k)$-t, akkor (q, k)-n átmegy az optimális megoldás.

Mivel egy adott $opt(i, j)$ kiszámításához csak az $opt(i-1, j-1)$, $opt(i-1, j)$, illetve $opt(i, j-1)$ értékekre van szükségünk, ezért egy n-szer 2-es tömbben tárolhatjuk az eredményeket, mindig felülírva a már nem használandó oszlopot. g-re hasonló

igaz. Ilyen módon kiszámíthatjuk $opt(i, n/2), g(i, n/2)$, majd $opt(i, n/2) + g(i, n/2$ értékét $\forall i \in \{0, 1, \ldots, m\}$-re. Kiválasztjuk ezek minimumát, ami tehát egy optimális megoldásban szerepelni fog. Utána ugyanezt elvégezzük $X = (x_1, \ldots, x_i)$ és $Y = (y_1, \ldots, y_{n/2})$, illetve $X = (x_i, \ldots, x_n)$ és $Y = (y_{n/2}, \ldots, y_m)$ karaktersorozatokra. Ilyen módon egy lineáris tárigényű rekurzív algoritmust kapunk.

(Bővebben, bizonyításokkal és pszeudokóddal lásd: [4].)

4.7. Tükörszó

Feladat: Adott egy karaktersorozat, és szeretnénk meghatározni karakterbeszúrásokkal tükörszóvá alakításának minimális költségét.

Jelölje $s(x)$ az x. karaktert, $p(x, y)$ az x. karaktertől az y. karakterig a tükörszóvá alakítás költségét.

$s(x) = s(y)$ esetén érvényes a $p(x, y) = p(x + 1, y - 1)$ összefüggés, mivel a karaktersorozat két vége már egyezik, csak a közepét kell tükörszóvá alakítani.

$s(x) \neq s(y)$ esetén $p(x, y) = min\{p(x + 1, y) + 1, p(x, y - 1) + 1\}$, mivel az utolsó karaktereknek az átalakítás után meg kell egyezniük, tehát valamelyik végére be kell szúrnunk egy új karaktert, és a maradék közbülső részt is tükörszóvá kell alakítanunk.

$p(x, y)$-t $|x-y|$ szerint növekvő sorrendben tudjuk kiszámítani, mert mindig csak a nála kisebb karakterrészletek optimumát használjuk fel.

A feladat visszavezetése karaktersorozatok hasonlóságára

A csere költségét válasszuk olyan magasnak, hogy semmiképp ne érje meg azt használni, például ha a tükörszóvá alakítandó szó hossza n, akkor a csere költsége legyen $5 * n$. A beszúrás költségét válasszuk 1-nek. Az X karaktersorozat legyen a tükörszóvá alakítandó karakterláncunk, az Y karaktersorozat pedig a tükörszóvá alakítandó karakterlánc fordított sorrendben felírva. Ezek optimális egymásba alakítása megadja az eredeti feladat egy optimális megoldását. (Bár a karaktersorozatok

összehasonlítása során kétszerannyi beszúrást használunk.)

Ugyanis a két karakterlenc első, illetve utolsó karakterei pontosan akkor egyeznek meg, tehát pontosan akkor nem szúrunk oda be új karaktert, ha a kérdéses szó első és utolsó karaktere megegyezik, vagyis amikor már csak a belsejét kell tükörszóvá alakítani. Ha pedig az első, illetve utolsó karakter különböző, akkor mind a tükörszóvá alakításban, mind a karakterek egymásba alakításánál szükség van beszúrásra. Ez azt jelenti, hogy a szó első és utolsó karakterét elhagyva egy ugyanilyen típusú, kisebb karakterszámú feladatot kapunk, amikre hasonlóan el lehet ugyanezt a gondolatmenetet mondani. Végül elérkezünk vagy egy egyetlen karakterből álló, vagy egy két karakterből álló szóhoz. Ezekre már triviális, hogy a tükörszóvá alakítás költsége, valamint a karaktersorozatok egymásba alakítása ekvivalens feladat. (Egy karakter, illetve két egyező karakter esetén egyiknél sem kell tenni semmit, két különböző karakter esetén pedig a tükörszóvá alakításhoz egy karakter, a karaktersorozatok egyezéséhez két karakter beszúrására van szükség.)

Irodalomjegyzék

[1] A. Frank, *Operációkutatás, jegyzet*

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Algoritmusok*, Műszaki Könyvkiadó (2003), 259-282.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Új algoritmusok*, Scolar Informatika (2003), 288-325.

[4] J. Kleinberg, É. Tardos, *Algorithm Design*, Pearson International Edition (2006), 251-336.

[5] J. Feldman, M. Ruhl, *Optimal Roundtrip Paths*, (1999)

[6] T. Király, L. Szegő, *Kiegészítés az Egészértékű Programozás I. és II. tárgyhoz, jegyzet*

[7] Z. Kátai, *Algoritmusok felülnézetből*, Scientia Kiadó (2007), 138-209.

[8] R. M. Karp, *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations (1972), 86-103.

[9] S.E.Dreyfus, R.A. Wagner, *The Steiner problem in graphs*, Networks (1972)