

1. Az alapötlet

A dinamikus programozás alapötlete egyszerű. Az alapfeladatot egy nagyobb „feladat-sokaságba” ágyazzuk. Az alapfeladat a sokaság egyik feladata. A sokaságnak azonban lesznek nagyon egyszerű esetei, amelyek megválaszolása senkit sem állít kihívás elé. A sokaság feladatainak lesz egy természetes sorrendje (ez lehet rendezés, de lehet részben rendezés is). A feladatok sorrendben való elvégzése egyszerű lesz. A korábban megoldott feladatokra alapozva az elsőre nehéz kérdések is könnyen megválaszolhatók lesznek.

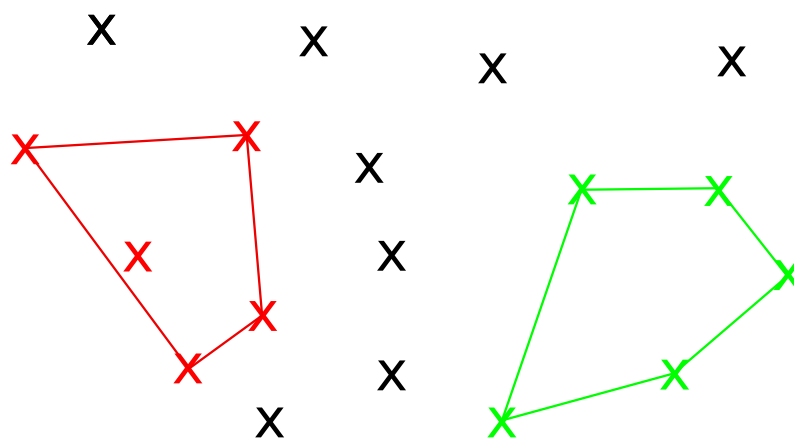
A kihívás a jó feladat-sokaság megválasztása. Ha ezt jól választottuk meg, akkor feladataink „felgöngyölítése” gyakran rutinból megoldható. Egy algoritmikus probléma-sereg dinamikus programozással történő megoldása nagy hasonlóságot mutat egy állítás-sereg teljes indukciós bizonyításával.

Formális leírás helyett nézzünk példákat.

2. Példák

I)

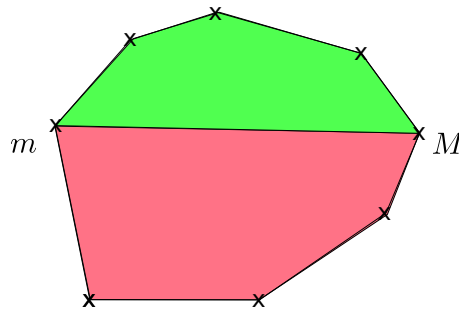
Legnagyobb konvex helyzetű részhalmaz megkeresése egy véges síkbeli ponthalmazban. Adott n darab pont a síkon: $\mathcal{P} = \{P_i(x_i, y_i)\}_{i=1}^n$. Feltesszük, hogy pontjaink x -koordinátái különbözőek. Határozzuk meg azt a legnagyobb \mathcal{R} részhalmazát, amelyek konvex helyzetűek, azaz egy konvex $|\mathcal{R}|$ -szög csúcshalmaza.



1. ábra. Egy véges ponthalmaz és egy öt elemű része (piros), amely nem konvex helyzetű, továbbá egy másik öt elemű része (zöld), amely konvex helyzetű.

Először is ponthalmazunkat balról jobbra végigpásztázzuk. Pontosabban az x_i koordinátákat sorbarendezzük. Az egyszerűség kedvéért feltesszük, hogy inputunk már ilyen, azaz $i < j$ esetén $x_i < x_j$. $\mathcal{O}(n \log n)$ összehasonlítással megoldható ez a része algoritmusunknak.

Másodszor is k konvex helyzetű csúcs konvex burka egy konvex k -szög. Ennek kerületének két pontját kiemeljük: a minimális és a maximális x koordinátájú csúcsot (m és M). Ez a kerületet, mint körszerűen rendezett véges csúcshalmazt két „ívre” osztja (a két kiemelt csúcsot mindkettőhöz hozzáértjük): egy felsőre és egy alsóra. A felső ív csúcsai is konvex helyzetűek. A konvex burkukhoz minden pont hozzájárul mint csúcs. Sőt, mM egy oldal lesz és az egész konvex burok ezen oldal felett fekszik. Azt mondjuk a felső ív egy *sapka* (angolul cap). Hasonlóan kezelhető az alsó ív. Erre azt mondjuk, hogy csésze (angolul cup).



2. ábra. Egy konvex helyzetű ponthalmaz, a kiemelt két csúccsal. A zöld rész a sapka összetevő konvex burka, a piros a csésze összetevő konvex burka.

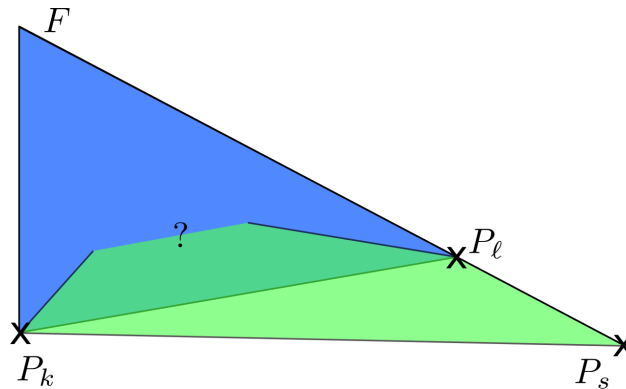
Ha minden $i < j$ esetén tudjuk az P_i -vel kezdődő és P_j -vel végződő leghosszabb sapka, illetve csésze méretét, akkor az alapkérdést is megválaszoljuk. Ez az ötlet már egy „dinamikus programozásos” ötlet. Ennek ellenére ezen a ponton csak arra használjuk, hogy az alapfeladatot „elfelezzük”, csak a legnagyobb sapkaméret meghatározását tárgyaljuk adott P_i és P_j pontok között. Természetesen az alábbi ötletek a legnagyobb csészeméret meghatározásához is használhatók. Így az eredeti feladatot megoldó algoritmus is kiolvasható belőle.

Az első két ötlet után megoldjuk a P_i -től P_j -be vezető leghosszabb sapka meghatározásának problémáját. A jelölés leegyszerűsítése miatt feltesszük, hogy $i = 1$ és $j = n$.

Legyen $\mathcal{F}_{k,\ell,s}$ ($k \leq \ell < s$) az a feladat, amely megadja a leghosszabb P_k -val kezdődő, P_s -ben végződő, oda közvetlenül P_ℓ -ből jutó sapka problémáját. Így $\mathcal{O}(n^3)$ részfeladatot definiáltunk. Ezek felgöngyöltése $\ell - k$ szerint növekvő sorrendben kezdjük. A legegyszerűbb/kezdő problémák az $\mathcal{F}_{k,k,s}$ problémák. Ekkor a P_k, P_s két pont egy sapkát alkot, ami az optimális sapka.

Tegyük fel, hogy $k < \ell$ és $\mathcal{F}_{k,\ell,s}$ problémával foglalkozunk, de az összes $\mathcal{F}_{k_0,\ell_0,s_0}$ probléma megoldása ismert, ha $\ell_0 - k_0 < \ell - k$. Csak azok a feladatok érdekesek, ahol a P_k, P_ℓ, P_s hármas sapkát alkot (ez nem volt probléma a $K = \ell < s$ esetben). Ezt a továbbiakban feltesszük.

Tippeljük meg milyen csúcs lehet a sapka hátulról harmadik (azaz a P_ℓ előtti) pontja. Ez a csúcs geometriailag egy háromszögből kerül ki: A P_k -n átmenő függőleges egyenestől balra kell lennie a sapka hiányzó részének. A $P_k P_\ell$ egyenes felett és a $P_\ell P_s$ egyenes alatt. A kijelölt háromszög határán nem feket a csúcsunk, de kivételt alkot P_k , ami előfordulhat, mint közvetlenül P_ℓ előtt álló csúcs.



3. ábra. F a P_i -n áthaladó függőleges egyenes és a $P_\ell P_s$ egyenes metszéspontja. A kék $P_i P_\ell F$ háromszögben kell megkeresni az optimális sapkát.

Vesszük $P_k P_\ell F$ háromszögbe eső pontjainkat (ezek P_k és még esetleg további belső pontok a háromszögből). Mindegyik P_ℓ -pontra a korábbi munkánkból kiolvassuk $\mathcal{F}_{k,\ell^-,\ell}$ megoldását. Ezekből a számokból a legnagyobbhoz 1-et hozzáadva kapjuk $\mathcal{F}_{k,\ell,s}$ megoldását.

Fontos, hogy az $\mathcal{O}(n^3)$ feladat közül a már megoldottak megoldását tároljuk és ebből olvassuk ki ami szükséges számunkra.

Megjegyzés. Kezdetben a dinamikus programozást a teljes indukcióhoz „hasonlítottuk”. Igazából egy rekurzióról van szó. Egy rekurzió gyakran nagyon egyszerűen leprogramozható. Lényeges különbség van a dinamikus programozás és emögött lévő rekurziós gondolat programozásilag leírt rekurzív kódolása között.

Ha a Fibonacci-számokat az

```
Fibonacci(n): if n>2 return Fibonacci(n-1)+ F(n-2) else return 1;
```

programmal számoljuk ki, akkor a `Fibonacci(10)` hívás után az nem foglal le egy 10 hosszú tömböt a szükséges Fibonacci-számok tárolására. Egy bonyolult gépi szintű feloldás történik.

A dinamikus programozás pedig éppen egy 10 hosszú tömbbel dolgozik. A futás végén mind a 10 szám ott lesz a memóriában.

Emiatt a rekurzív program futása lassabb lesz (sok újra számolás történik). A dinamikus programozási algoritmusok sebessége attól függ hány részfeladatot tartalmaz feladat-seregünk.

A dinamikus programozás formailag gyakran úgy jelentkezik, hogy egy számsorozatot, vagy egy számtáblázatot, vagy bonyolultabb szám-sereget töltünk ki meghatározott sorrendben.

II)

A legnagyobb független ponthalmaz megkeresése egy fában. Adott egy T fa. Határozzuk meg a legnagyobb független halmazának méretét.

Emléztetőnek: egy fa egy körmentes, összefüggő gráf. Egy F csúcshalmaz független, ha nincs olyan él amely mindkét végpontja.

Először jelöljük ki egy speciális $r \in V(T)$ csúcst, amit gyökérnek nevezünk. A (T, r) gyökeres fa új elnevezéseket indukál. A csúcshalmaz diszjunkt osztályokba sorolható a gyökértől vett távolság szeint. Legyen G_i a gyökértől i távolságra lévő

csúcsok halmaza. $G_0 = \{r\}$, G_1 a gyökér szomszédait tartalmazó csúcshalmaz. G_2 a gyökér másodsomszédainak halmaza. Minden $e \in E(T)$ él esetén van olyan $i \in \mathbb{N}$, hogy e az $x \in G_i$ és $y \in G_{i+1}$ csúcsokat kösse össze. Ebben az esetben x az e élnek megfelelő szülő csúcs, míg y a gyerek csúcs. Azt mondjuk, hogy x gyereke y , y -nak x a szülője. Ha x nem a gyökér, akkor x tehetünk egy egyértelmű lépést a gyökér felé. Ezzel x egyetlen szülőjébe (gyakori az *apa* elnevezés, ekkor a gyerekekre a *fiú* egy alternatív terminológia) jutunk. A gyökér a fa *ősapja*, ennek nincs szülője. Ha egy csúcsnak nincs gyereke, akkor *levélnék* nevezzük.

G_1 tartalmazza a gyökér gyerekeit/fiait, G_2 tartalmazza a gyökér unokáit.

Ha y -ból a gyökér felé haladva elérjük az x csúcsot, akkor x az y csúcs felmenője, y az x leszármazottja. Az r gyökér minden más csúcsnak a felmenője. Egy ℓ levélcúcs esetén ℓ -nek nincs leszármazottja. Egy x csúcs meghatároz/„generál” egy részfát, amely csúcsai x és leszármazottjai, élei a T fa ezen csúcsai között haladó élei. Ezt a részfát T_x -szel jelöljük és egy x -ben gyökereztetett részfának gondoljuk. Egy n csúcsú fának n darab csúcs által generált rész gyökeres fája van.

Egy gyökeres fa mélysége a leghosszabb gyökér-levél út hossza. A 0 mélységű gyökeres fa egyetlen csúcsból áll, a gyökérből, ami egyben levélcúcs is.

Megjegyzés. A fenti szleng/terminológia onnan ered, hogy egy gyökeres fára egy családfa a legjobb példa. Az Árpád-ház családfájában a fenti elnevezések mindennapi értelmet kapnak.

Megjegyzésünk után az alapfeladatra úgy gondolunk, hogy inputja egy (T, r) gyökeres fa. A független halmazok fogalmát a gyökér nem befolyásolja, csupán kibővíti „nyelvünket”.

A feladat-sereg $\{\mathcal{F}_x\}_{x \in V(T)}$ lesz, ahol \mathcal{F}_x a (T_x, x) gyökeres fára vonatkozó feladat, a legnagyobb független csúcshalmaz méretének megválaszolása. Tehát $|V(T)|$ darab részfeladatot definiáltunk. A feladatok felgöngyölítési sorrendje a mélység szerinti növekvő sorrend.

Azaz azokkal a feladatokkal kezdjük, ahol (T_x, x) 0 mélységű, azaz x az eredeti fa egy levele. Ekkor egy egy csúcsú, 0 élű gráfunk van, amelyben $\{x\}$ egyben független halmaz is (az \emptyset mellett), azaz a legnagyobb független halmaz mérete 1.

Tegyük fel, hogy feladataink között (T_x, x) a soros és a kisebb mélységű gyökeres fákra vonatkozó problémákat már megoldottuk. A független halmazokat osszuk két osztályba. Az x -et nem tartalmazók ((a) eset), és az x -et tartalmazók ((b) eset).

- (a) Legyen f_1, \dots, f_d az x csúcs fiai. Ekkor $T_{f_1}, T_{f_2}, \dots, T_{f_d}$ részfákban tetszőlegesen/függetlenül vehetünk egy független halmazt és ezek uniója egy (a) eset alá tartozó független halmazt ad. Másrészt minden (a) eset alá tartozó független halmazt megkapunk így. A független választások miatt egyszerű az elemszám maximalizálása. Mindegyik T_{f_i} -ből egy legnagyobb méretű független halmazt kell választanunk. Ezek a méretek rendezésünkre állnak, mert (T_x, x) mélysége nagyobb mint (T_{f_i}, f_i) mélysége. Ezek összege az aktuális feladatra adott válasz.
- (b) Legyen u_1, \dots, u_D az x csúcs unokái. Ekkor $T_{u_1}, T_{u_2}, \dots, T_{u_D}$ részfákban tetszőlegesen/függetlenül vehetünk egy független halmazt és ezek uniója, x -szel kibővítve egy (b) eset alá tartozó független halmazt ad. Másrészt minden (b) eset alá tartozó független halmaz előáll így. A független választások miatt egyszerű az elemszám maximalizálása. Mindegyik T_{u_i} -ből egy legnagyobb

méretű független halmazt kell választanunk. Ezek a méretek rendezésünkre állnak, mert (T_x, x) mélysége nagyobb mint (T_{u_i}, u_i) mélysége. Ezek összege „meg egy” az aktuális feladatra adott válasz.

Mindkét esetben meghatározhatjuk a megfelelő típusú független halmazok maximális méretét. A két lehetőség lefedi az összes független halmazt, tehát a két maximum közül a nagyobbik adja algoritmusunk outputját.

Megjegyzés. Könnyű mindkét algoritmust átlakítani úgy, hogy ne csak egy maximális méretet adjon meg, de egy legnagyobb konvex helyzetű részhalmaz, vagy egy legnagyobb független csúcshalmaz legyen az output.