Alapvető algoritmusok

10. előadás

Dr. Pătcaș Csaba

BABEȘ-BOLYAI TUDOMÁNYEGYETEM
Matematika és Informatika Kar

Tartalom

Autó bérbeadás

Feladat

Egy szállítási vállalat autókat kölcsönöz. Egy bizonyos jármű iránt igen nagy az érdeklődés, ezért az igényeket egy évre előre jegyzik. Az igényt két számmal jelöljük, amelyek az év azon napjainak sorszámait jelölik, amellyel kezdődően, illetve végződően igénylik az illető autót. Állapítsuk meg a bérbeadást úgy, hogy a lehető legtöbb személyt szolgáljuk ki!

Példa: $n = 10$
[1 23], [12 20], [5 10], [12 29], [13 25], [40 66], [30 35], [22 33], [70 100], [19 65]
Egy lehetséges optimális megoldás: [5 10], [12 20], [22 33], [40 66], [70 100]
Egy másik optimális megoldás: [5 10], [13 25], [30 35], [40 66], [70 100]

Autó bérbeadás

Megoldás

- Az első gondolatunk az lehetne, hogy prioritással válasszuk ki azokat az intervallumokat, amelyek a legrövidebbek, legkorábban kezdődnek, vagy esetleg legkevesebb más intervallumot metszenek.
- Ezek a megközelítések viszont nem vezetnek maximális megoldáshoz, a megfelelő ellenpéldák megkeresése jó gyakorlat és melegen ajánlott mindenkinek egyénileg.
- A helyes megoldás az intervallumok végpont szerinti rendezése és ebben a sorrendben bejárva őket, mindig az első lehetséges intervallum hozzáadása a megoldáshoz.
- A pszeudokódban feltételezzük, hogy az a tömb struktúrákat tartalmaz, melyeknek a `kezd` és `vég` mezői tárolják az intervallumok kezdő- és végpontjait.

Autó bérbeadás
Pszeudokód

```
ALGORITMUS Autó(n, a, megoldás)
  Rendez(n, a)
  Hozzáfűz(megoldás, a[1])
  utolsó = a[1]
  MINDEN i = 2, n végezd el:
    HA (a[i].kezd >= utolsó.vég) akkor
      Hozzáfűz(megoldás, a[i])
      utolsó = a[i]
    VÉGE(Ha)
  VÉGE(Minden)
VÉGE(Algoritmus)
```

A hátizsák folytonos feladata

Feladat

Egy tolvaj betört egy hentesüzletbe, ahol n áru közül válogat. Minden árunak ismeri a súlyát és az értékét. Mivel a hátizsákjába legtöbb S súly fér, szeretne úgy válogatni, hogy a nyeresége maximális legyen. A tolvaj tetszőlegesen darabolhatja az árukat, de ilyen esetben az áru értéke a súlyával arányosan csökken.

A feladatot a szakirodalomban töredékes hátizsák, vagy folytonos hátizsák feladatként találhatjuk meg.

Példa: $n = 4, S = 10$

Súlyok: [5 4 5 2]

Értékek: [10 20 1 3]

Az optimális össznyereség 31.5, amelyet az első és a második áruk teljes egészében való kiválasztásából és a negyedik árú felének kiválasztásából kapunk.

A hátizsák folytonos feladata
Megoldás

- Az első ötlet lehetne a legértékesebb áruk, esetleg a legkönnyebb áruk kiválasztása.
- Ezek a megközelítések nem vezetnek mindig optimális megoldáshoz, az ellenpéldák megkeresése mindenkinek egyéni feladat.
- A helyes megoldás, hogy a tárgyakat az *érték* / *súly* arány szerint csökkenő sorrendbe rendezzük.
- Ezt a sorrendet bejárva, addig teszünk be a hátizsákba teljes tárgyakat, amíg ez lehetséges.
- Az ezeket követő tárgynak csak egy darabját választjuk ki, ha van erre lehetőség.
- A pszeudokódban az `arány[i]` értékbe azt mentjük el, hogy a rendezés utáni sorrendben vett i. áru hányad részét választjuk bele a megoldásba. Tehát ez az érték akkor lesz 1, ha a teljes árut a hátizsákba tesszük és 0, ha nem választjuk ki.

A hátizsák folytonos feladata
Pszeudokód

```
ALGORITMUS FolytonosHátizsák(n, S, súly, érték, arány)
  RendezCsökkenőbe(n, súly, érték)
  MINDEN i = 1, n végezd el:
    arány[i] = 0
  VÉGE(Minden)
  hely = S
  i = 1
```

A hátizsák folytonos feladata

Pszeudokód

```
  AMÍG ((i <= n) ÉS (hely > 0)) végezd el:
    HA (súly[i] <= hely)
      arány[i] = 1
      hely = hely - súly[i]
    KÜLÖNBEN
      arány[i] = hely / súly[i]
      hely = 0
    VÉGE(Ha)
    i = i + 1
  VÉGE(Amíg)
VÉGE(Algoritmus)
```

Növekvő részsorozatokra bontás

Feladat

Adott egy n elemű természetes számokból álló számsorozat. Bontsuk fel minimális számú szigorúan növekvő részsorozatra!

Példa: $n = 8$, [5 1 6 2 8 5 7 4]

[5 6 8]

[1 2 5 7]

[4]

Növekvő részsorozatokra bontás

- Optimális megoldást kaphatunk a következő mohó stratégiával.
- Balról jobbra vesszük sorra a számokat.
- Ha egy számot nem írhatunk egyik épülő részsorozat végére sem, akkor új részsorozatot kezdünk.
- Ha van olyan részsorozat, amelynek a végére írhatjuk az aktuális számot, akkor azt választjuk közülük, amelyik végén a legnagyobb szám található.
- Intuitíven ez a választás biztosítja, hogy a végződések között a legkisebb értékek maradjanak meg.
- A fenti példában a 6-ost írhattuk volna az 1-es után is, de ekkor a végződések [6 1] helyett [5 6]-ként alakultak volna, így a 2-essel új sorozatot kellett volna kezdenünk.
- Hasonlóképpen, a 8-ast írhattuk volna a 2-es után is, de ekkor a végződések [8 2] helyett [6 8] értékeket vettek volna fel és az 5-össel új sorozatott kellett volna kezdenünk.

Növekvő részsorozatokra bontás

Hogyan implementáljuk hatékonyan?

- Legegyszerűbben implementálva a megoldást egy mátrixban tárolnánk és lineárisan keresnénk meg a lehetséges végződések közül a legnagyobbat, így a memóriabonyolultság és a legrosszabb esetben az időbonyolultság is $\Theta(n^2)$ lenne.
- A megoldást tárolhatjuk egy `következő` nevű vektorban is, amely minden elemre tárolja, hogy melyik indexű elem következik utána a megfelelő részsorozatban, vagy -1-et, ha utolsó a saját részsorozatában. A fenti példában `következő` $=$ [3 4 5 6 -1 7 -1 -1]

Növekvő részsorozatokra bontás

Hogyan implementáljuk hatékonyan?

- Vegyük észre, hogy a végződések sorozata (fentről lefele nézve) minden lépésben csökkenő sorozatot alkot. Ez lehetővé teszi, hogy ne lineárisan, hanem binárisan keressük meg az aktuális szám helyét. (A bináris keresésről bővebben az *Oszd meg és uralkodj* módszernél lesz szó).
- Tehát elég a `következő` sorozatot és a `végződések` sorozatát tárolni, így a memóriabonyolultságot $\Theta(n)$-re csökkentettük.
- A bináris keresést alkalmazva az időbonyolultság $\Theta(n \log n)$ lesz a legrosszabb esetben.

Bónusz

Feladat

Adottak n és s természetes számok ($1 \leq n \leq 1000, |s| \leq 1000000$). Elhelyezhetjük-e a $+$ és - műveleti jeleket az $1, 2, \ldots, n$ számok közé (a számok sorrendjét megtartva), úgy, hogy az eredmény s legyen? Írjunk ki egy tetszőleges megoldást, vagy -1-et, ha nem létezik ilyen!

Példa:
$n = 9, s = 5$
$1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9$
Mennyi? 2 pont a H2 jegybe
Hova? Canvas privát üzenethez csatolva a forráskódot, amelyben kommentként szerepel a megoldási ötlet
Hányszor? Diákonként csak egyszer lehet beküldeni megoldást
Meddig? December 21., 23:59

Tartalom

- Egy gráfnak vannak csúcsai és élei. Irányított gráfokban az irányított éleket gyakran íveknek is nevezzük.
- Egy gráfot elképzelhetünk úgy, mint városok, melyeket utak kötnek össze.
- A továbbiakban a csúcsok számát n-el, az élek számát m-el jelöljük.
- Az élek lehetnek irányítatlanok és irányítottak.
- Egy nem irányított gráf összefüggő, ha bármely csúcsból el lehet jutni bármely másik csúcsba.
- A gráf lehet súlyozott vagy nem súlyozott.

- Egy gráf részgráfját úgy kapjuk, hogy kitörlünk az eredeti gráfból tetszőleges számú (akár 0) csúcsot és élet.
- Ha csak éleket törlünk, akkor beszélünk feszítő részgráfról.
- A fa egy összefüggő gráf, amelyben nincsenek körök.
- Egy gráf feszítőfája egy olyan feszítő részgráf, amely fa.
- Megjegyezzük, hogy feszítőfája csak összefüggő gráfoknak van.

Minimális feszítőfa

Feladat

Adott egy n csúcsú és m élű nem irányított, súlyozott gráf. Határozzuk meg a gráf minimális feszítőfáját, vagyis azt a feszítőfát, amelyhez tartozó élek költségeinek az összege a lehető legkisebb. Ha létezik több megoldás, elég egyet meghatározni.

- A minimális feszítőfa megkeresése egy klasszikus gráfelméleti feladat.
- Számos megoldási algoritmus ismert rá, ezek közül a legfontosabbak: Jarník–Prim–Dijkstra algoritmus, Kruskal algoritmusa és Borůvka algoritmusa.
- Mindhárom algoritmus a mohó módszert használja.
- Ezeket többféleképpen lehet implementálni, speciális adatszerkezetek felhasználásával javítható a bonyolultságuk.

Kruskal algoritmusa

- Lássuk Kruskal algoritmusának egy egyszerűen implementálható változatát.
- Rendezzük az éleket költségeik szerint növekvő sorrendbe.
- Ebben a sorrendben egyenként adjuk hozzá őket a megoldáshoz, kivéve, ha egy él hozzáadása kört hozna létre (mert akkor már nem beszélhetnénk fáról).
- Mivel egy n csúcsú fának $n-1$ éle van, megállhatunk amikor $n-1$ élet adtunk hozzá a megoldáshoz.

Példa: $n = 5, m = 7$
Élek: (3, 4), (1, 2), (1, 3), (2, 4), (4, 5), (2, 5), (3, 5)
Költségek: 1, 2, 3, 4, 5, 6, 7

Kruskal algoritmusa

- Az algoritmus kritikus pontja, annak az ellenőrzése, hogy egy él hozzáadása kört hoz-e létre.
- Ez akkor áll fenn, ha az él két végpontja már azonos összefüggő komponenshez tartozik.
- Ennek a legegyszerűbb ellenőrzése egy mélységi vagy szélességi bejárással lehetséges, így az algoritmus végső bonyolultsága $O(m \cdot n)$ lenne.
- Ezen tudunk javítani, ha egy `komp` tömbben tároljuk, hogy melyik csúcs melyik sorszámú komponenshez tartozik.
- Kezdetben mindegyik csúcs izolált, vagyis önállóan alkot egy komponenst.
- Egy él hozzáadásakor a két komponenst összefűzzük.
- Az `élek` tömb mindegyik eleme egy struktúra, melynek `u` és `v` mezői az adott él két végpontját, a `költség` mező meg a súlyát tárolja.

Kruskal algoritmusa

```
ALGORITMUS Kruskal(n, m, élek, feszítőfa)
  RendezNövekvőbe(m, élek)
  MINDEN i = 1, n végezd el:
    komp[i] = i
  VÉGE(Minden)
  i = 0
  j = 1
```

Kruskal algoritmusa

```
AMÍG ((j <= m) ÉS (i < (n - 1))) végezd el:
  HA (komp[élek[j].u] != komp[élek[j].v]) akkor
    i = i + 1
    Hozzáfűz(feszítőfa, élek[j])
    id = komp[élek[j].v]
    MINDEN k = 1, n végezd el:
      HA (komp[k] = id)
        komp[k] = komp[élek[j].u]
      VÉGE(Ha)
    VÉGE(Minden)
  VÉGE(Ha)
  j = j + 1
VÉGE(Amíg)
VÉGE(Algoritmus)
```

Dijkstra algoritmusa

Feladat

Adott egy n csúcsú m élű irányított, súlyozott gráf. Határozzuk meg a legrövidebb utak hosszát, amelyek egy adott s csúcsból indulnak ki a gráf összes többi csúcsa felé és írjuk ki ezeket az utakat. Egy út hossza egyenlő az éleihez rendelt költségek összegével.

- Az egy csúcsból kiinduló legrövidebb utak megkeresésére Dijkstra algoritmusát alkalmazzuk, amely ugyanúgy működik irányított és irányítatlan gráfokra is.
- Az algoritmus csak nemnegatív költségek esetén ad helyes megoldást.

Dijkstra algoritmusa

- A legrövidebb utak tulajdonképpen egy gyökeres fát fognak alkotni, így egyetlen `p` tömb segítségével tárolhatjuk ezeket: a `p[i]` tárolja, hogy melyik csúcs előzi meg az i csúcsot a hozzá vezető legrövidebb úton (az s-ből indulva).
- A `d` tömbben tároljuk mindegyik csúcshoz vezető legrövidebb utak hosszát.
- Szükségünk van egy `ismeretlen` halmazra, ebben tároljuk azokat a csúcsokat, amelyekbe még nem ismerjük a legrövidebb út hosszát az s kiindulási csúcsból.
- A mohó módszer elvét követve, mindig azt a csúcsot vesszük ki az `ismeretlen` halmazból, melynek a legkisebb érték felel meg a `d` tömbben.
- Minden lépésben, a kiválasztott csúcson keresztül próbáljuk javítani az utakat azokba a csúcsokba, amelyek még az `ismeretlen` halmazban vannak.
- A csúcsok többet nem kerülnek vissza az `ismeretlen` halmazba.
- A gráfot az a adjacencia mátrixban kapjuk meg, vagyis `a[i][j]` az i és j közötti ív költségét tárolja, vagy `végtelen`-t, ha nincs él i és j között.

$n = 7, m = 10$

Ív	Költség
(1, 2)	10
(1, 3)	4
(1, 4)	9
(3, 2)	2
(2, 5)	5
(3, 5)	30
(4, 6)	5
(5, 7)	7
(3, 6)	15
(6, 7)	2

Dijkstra algoritmusa
Pszeudokód

```
ALGORITMUS Kezdőértékek(n, a, s, ismeretlen, d, p)
  Minden i = 1, n végezd el:
    d[i] = a[s][i]
    ismeretlen[i] = IGAZ
    p[i] = s
  VÉGE(Minden)
  d[s] = 0
  ismeretlen[s] = HAMIS
VÉGE(Algoritmus)
```

Dijkstra algoritmusa

Pszeudokód

```
ALGORITMUS d_min(n, ismeretlen, d, ind)
  min = végtelen
  MINDEN j = 1, n végezd el:
    HA (ismeretlen[j]) akkor
      HA (d[j] < min) akkor
        min = d[j]
        ind = j
      VÉGE(Ha)
    VÉGE(Ha)
  VÉGE(Minden)
  VISSZATÉRÍT: min
VÉGE(Algoritmus)
```

Dijkstra algoritmusa

Pszeudokód

```
ALGORITMUS Dijkstra(n, a, d, p)
  AMÍG (d_min(n, ismeretlen, d, ind) < végtelen)
    ismeretlen[ind] = HAMIS
    MINDEN j = 1, n végezd el:
      HA (ismeretlen[j] ÉS (a[ind][j] < végtelen)) akkor
        dd = d[ind] + a[ind][j]
        HA (dd < d[j])
          d[j] = dd
          p[j] = ind
        VÉGE(Ha)
      VÉGE(Ha)
    VÉGE(Minden)
  VÉGE(Amíg)
VÉGE(Algoritmus)
```

Tartalom

A hátizsák diszkrét feladata

- Láttuk korábban a hátizsák folytonos feladatát.
- A diszkrét változatban a tárgyakat nem szabad darabolni.
- Ha a folytonos változatnál látott megoldást használjuk heurisztikus megközelítésként, ez nem ad mindig helyes eredményt.
- Ellenpélda: $n = 3, S = 5$
  Súlyok: [4 3 2]
  Értékek: [6 4 2.5]
- Az *érték* / *súly* arányok a következők: [1.5 1.33 1.25]
- A hátizsákba csak az első tárgy kerülne, ezzel az összérték 6 lenne.
- Az optimális megoldás a második és a harmadik tárgy kiválasztása 6.5-ös összértékkel.

Összegkifizetés minimális számú bankjeggyel

Feladat

Határozzuk meg egy módját az S pénzösszeg kifizetésének minimális számú bankjeggyel. A $b_1, b_2, \ldots, b_n$ címletű bankjegyek végtelen számban állnak rendelkezésünkre.

- A legkézenfekvőbb heurisztika, hogy a legnagyobb címlettel kifizetünk amennyit lehet, majd a második legnagyobb címlettel folytatjuk és így tovább.
- De így előfordulhat, hogy a fennmaradt összeget nem lehet kifizetni a még kiválasztatlan bankjegyekkel és így a greedy heurisztika nem találja meg az egyébként létező megoldást.

Összegkifizetés minimális számú bankjeggyel

Ellenpéldák

$S = 12, b_1 = 5, b_2 = 4, b_3 = 1$
Optimális megoldás (összesen 3 bankjegy): $12 = 3 \cdot 4$
Greedy heurisztikával (összesen 4 bankjegy): $12 = 2 \cdot 5 + 2 \cdot 1$

$S = 12, b_1 = 5, b_2 = 4, b_3 = 3$
Optimális megoldás (összesen 3 bankjegy): $12 = 3 \cdot 4$
Greedy heurisztikával (nem talál megoldást): $12 = 2 \cdot 5 + \ldots$

Gráfszínezés

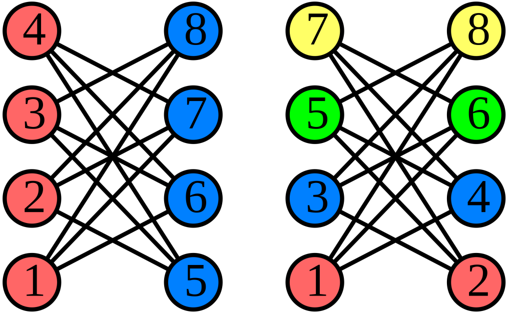
Feladat

Legyen egy n csomópontú nem irányított gráf, amelynek adottak az élei. Határozzuk meg a csomópontok legkevesebb színnel való kiszínezését úgy, hogy bármely két szomszédos csomópont különböző színű legyen.

- Ez egy klasszikus NP-teljes gráfelméleti feladat, így optimális megoldást csak exponenciális megoldással kaphatunk.
- A négyszín-tétel kimondja, hogy egy síkban ábrázolható gráfot (amelyet le tudunk rajzolni úgy, hogy az élei ne metsszék egymást), mindig ki lehet színezni négy színnel.
- Ez volt az első matematikai tétel, amelyet számítógép segítségével bizonyítottak.

Gráfszínezés

- A leggyakrabban használt heurisztika azon az elven működik, hogy bejárjuk sorban a gráf csúcsait és mindegyikhez a legkisebb olyan színt rendeljük, amit még nem használtunk fel a szomszédai között.
- A módszer eredményessége nagyban függ a csúcsok sorrendjétől, viszont mindig létezik egy olyan sorrend, amely optimális megoldáshoz vezet.



Csomópontok kizárása egy gráfból

Feladat

Legyen egy nem irányított gráf. Kizárás alatt azt a műveletet értjük, amellyel a gráfból törlünk egy csomópontot és vele együtt valamennyi szomszédját. Határozzuk meg azt a minimális számú kizárási műveletet, amellyel a gráf valamennyi csomópontja kitörölhető.

- A legnagyobb fokszámú csúcs kizárására alapuló heurisztika nem ad mindig helyes megoldást.
- A feladatra lehetséges nemdeterminisztikus heurisztikát is adni, amely véletlenszerű sorrendben zárja ki a csúcsokat.