

1 Teorija

Naslednja od metod v programiranju je dinamično programiranje. V splošnem spada sem več podmetod, tukaj pa bomo obravnavali le najbolj preprost primer. Pri požrešni metodi smo rešitev gradili sproti in se na vsakem koraku odločili, kaj gre v rešitev in kaj ne ter se nismo vračali (da bi iz rešitve odstranili del rešitve, ki ni perspektiven). Slabost takšnega pristopa je, da v splošnem ne vodi do optimalnih rešitev, ampak le približnih.

Da bi lahko prišli do optimalnih rešitev, zato ne tvorimo rešitev tako striktno, ampak si puščamo možnost, da vodimo več možnih rešitev in na koncu izberemo najboljšo. Med reševanjem torej vodimo seznam potencialnih rešitev, takšno potencialno rešitev pa takoj zavržemo, če ugotovimo, da ne vodi k rešitvi (npr. če polnimo nahrbtnik in ugotovimo, da seznam vstavljenih elementov že presega volumen nahrbtnika).

Pravimo, da dinamično programiranje rešujemo na podlagi pravila optimalnosti. Običajno si izpeljemo rekurzivno enačbo (t.i. Bellmanovo enačbo), ki jo rešujemo ali naprej ali nazaj.

Dinamično programiranje je močna strategija v reševanju problemov. Ključna ideja dinamičnega programiranja je, da velik problem razbijemo na manjše podprobleme, shranimo rešitve teh manjših podproblemov in, ko je potrebno, uporabimo shranjene rešitve pri reševanju originalnega problema. Na ta način se izognemo računanju iste količine znova in znova.

2 0/1 nahrbtnik z dinamičnim programiranjem

Na predavanju ste spoznali teoretični vidik reševanja tega problema, tukaj podajamo praktičnega, ki privede do optimalne rešitve. Problem rešimo v dveh fazah:

- gradnja rešitve, tako da vodimo vse pare (volumen, cena),
- rekonstrukcija rešitve.

Za bolj večje programiranja je ti dve fazi možno združiti v eno s primerno izbrano podatkovno strukturo.

Prva faza:

V prvi fazi gradimo možne rešitve in sproti zavračamo tiste, kjer je volumen večji od razpoložljivega. Zamislimo si zapis S_k kot množico parov (volumen, cena), ki predstavljajo možne rešitve tega problema, pri čemer začetna množica $S_0 = \{(0, 0)\}$ predstavlja prazen nahrbtnik. Zatem za vsak potencialni element ponovimo enak postopek: izračunamo vse pare (volumen, cena), kakor da bi ta element "vstavili" v nahrbtnik.

Pokažimo to na primeru: $n = 4$... število potencialnih elementov za v nahrbtnik

$V = 9$... volumen nahrbtnika

$v = (2, 4, 4, 6)$... volumni potencialnih elementov

$c = (3, 5, 7, 8)$... cene potencialnih elementov

Pričnemo z začetno množico $S_0 = \{(0, 0)\}$ in zatem, kakor da bi vstavili prvi element oz. par $(2, 3)$:

$$S_0 = \{(0, 0)\}$$

$$Z_1 = \{(0 + 2, 0 + 3)\} = \{(2, 3)\}$$

Vse elemente iz Z_1 prepíšemo v S_1 in pri istem volumnu zapišemo samo par z največjo ceno. Sledi naslednji element $(4, 5)$:

$$S_1 = \{(0, 0), (2, 3)\}$$

$$Z_2 = \{(0 + 4, 0 + 5), (2 + 4, 3 + 5)\} = \{(4, 5), (6, 8)\}$$

Vse elemente iz Z_2 prepíšemo v S_2 (dobro je, da so urejeni po velikosti cene). Tretji element za vstaviti je par $(4, 7)$:

$$S_2 = \{(0, 0), (2, 3), (4, 5), (6, 8)\}$$

$$Z_3 = \{(0 + 4, 0 + 7), (2 + 4, 3 + 7), (4 + 4, 5 + 7), (6 + 4, 8 + 7)\} = \{(4, 7), (6, 10), (8, 12), |(10, 15)\}$$

Par $(10, 15)$ je namensko zapisan za znakom $|$, ki označuje, da ta rešitev več ni primerna, ker je volumen vstavljenih elementov že večji od razpoložljivega. Po vstavljanju zadnjega para $(6, 8)$ dobimo:

$$S_3 = \{(0, 0), (2, 3), (4, 7), (6, 10), (8, 12)\}$$

$$Z_3 = \{(6, 8), (8, 11), | \dots \text{ vsi ostali } \dots \}$$

in na koncu S_4 , ki je v tem primeru slučajno tudi enak S_3 :

$$S_4 = \{(0, 0), (2, 3), (4, 7), (6, 10), (8, 12)\}$$

Druga faza:

V drugi fazi rekonstruiramo rešitev oziroma poiščemo kateri elementi bodo zares vstavljeni v nahrbtnik. Iz končne množice (v našem primeru S_4) izberemo par z največjo ceno, kar za naš primer znaša $(8, 12)$. Zatim "po poti nazaj" gledamo, kako smo prišli do tega para. Če par obstaja v S_k , ne pa tudi v S_{k-1} , smo do tega para zagotovo prišli tako, da smo dodali k -ti element. In če obstaja, odštejemo vrednosti tega elementa od para in ponovimo iskanje parov do začetne množice S_0 .

Nadaljevanje primera bi tako bilo:

1. Ker $(8, 12) \in S_4$ in $(8, 12) \in S_3$ dobimo, da četrti element $(6, 8)$ ni v optimalni rešitvi.
2. Ker $(8, 12) \in S_3$ in $(8, 12) \notin S_2$ dobimo, da je tretji element $(4, 7)$ v optimalni rešitvi. Odštejemo vrednost tega elementa in dalje iščemo par $(8 - 4, 12 - 7) = (4, 5)$.
3. Ker $(4, 5) \in S_2$ in $(4, 5) \notin S_1$, je drugi element $(4, 5)$ v optimalni rešitvi. Dalje iščemo par $(0, 0) \dots$

V optimalni rešitvi sta torej samo drugi in tretji element.

3 Dolžine vseh najkrajših poti

Podobno nalogo smo že reševali pri požrešni metodi, tukaj sledi izvedba z dinamičnim programiranjem, ki je precej enostavnejša za implementirati.

Naj bo d_{ij}^k enak dolžini najkrajše poti iz i v j , kjer vmesna vozlišča (postaje na poti) nimajo indeksa večjega od k . Torej, na poti od v_i do v_j so lahko samo vozlišča $v_{i+1}, v_{i+2}, \dots, v_{j-1}$.

Vse te cene lahko uredimo v matriko in obstaja jih natanko n^2 , kjer n označuje število vozlišč. Ker je graf lahko usmerjen, sta v splošnem d_{ij}^k in d_{ji}^k različna!

Ker velja

$$d_{ij}^k = \min\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\}, k = 1, 2, \dots, n; \forall i, j$$

$$D^k = (d_{ij}^k)_{i,j=1}^n$$

se da algoritem preprosto rešiti s spremembo matrik. Iščemo

$$D^n = (d_{ij}^n)_{i,j=1}^n$$

poznamo pa na začetku začetne cene povezav med posameznimi vozlišči (prvotna matrika cene povezav):

$$D^0 = (d_{ij}^0)_{i,j=1}^n$$

Sam zapis d_{ij}^k pomeni, da je dolžina najkrajše poti med i in j , preko vozlišč z indeksom največ k enaka najcenejši od poti brez vozlišča v_k ali pa preko tega vozlišča. Z drugimi besedami, če smo k povečali za 1 (torej v možna vozlišča na poti vključili še vozlišče v_k), potem je nova najcenejša pot samo, če sedaj preko vozlišča v_k dobimo manjšo ceno. V nasprotnem primeru pa ohranimo prejšnjo rešitev.

Postopek iskanja dolžin najkrajših poti na tak način je preprost algoritem, podan v naslednjem psevdokodu.

```

metoda DolzineNajkrajshihPoti(sttock, cenepovezav) {
  PripraviPrvotnoMatrikoDO(sttock, cenepovezav);
  for k ← 1 to n {
    for i ← 1 to n {
      for j ← 1 to n {
         $d_{ij} = \text{Min}(d_{ij}, d_{ik} + d_{kj});$  //i, j-ti element matrike
      }
    }
  }
} // konec metode

```

Sledi še primer. Podana je začetna matrika cen usmerjenega grafa, pri čemer ∞ pomeni, da ne obstaja direktna povezava med točkama.

$$D^0 = \begin{bmatrix} 0 & 10 & 31 & 5 \\ 6 & 0 & \infty & 11 \\ \infty & 9 & 0 & 22 \\ 20 & 14 & 18 & 0 \end{bmatrix}$$

Iščemo matriko D^4 , ki predstavlja rezultat, matriko D^1 pa dobimo na naslednji način:

$$\begin{aligned}
d_{11}^1 &= \dots \text{ ko je } i = j, \text{ je rezultat vedno } 0! \\
d_{12}^1 &= \min(d_{12}^0, d_{11}^0 + d_{12}^0) = \min(10, 0 + 10) = 10 \\
d_{13}^1 &= \min(d_{13}^0, d_{11}^0 + d_{13}^0) = \min(31, 0 + 31) = 31 \\
d_{14}^1 &= \min(d_{14}^0, d_{11}^0 + d_{14}^0) = \min(5, 0 + 5) = 5
\end{aligned}$$

$$\begin{aligned}
d_{21}^1 &= \min(d_{21}^0, d_{21}^0 + d_{11}^0) = \min(6, 6 + 0) = 6 \\
d_{23}^1 &= \min(d_{23}^0, d_{21}^0 + d_{13}^0) = \min(\infty, 6 + 31) = 37 \\
d_{24}^1 &= \min(d_{24}^0, d_{21}^0 + d_{14}^0) = \min(11, 6 + 5) = 11
\end{aligned}$$

$$\begin{aligned}
d_{31}^1 &= \min(d_{31}^0, d_{31}^0 + d_{11}^0) = \min(\infty, \infty + 0) = \infty \\
d_{32}^1 &= \min(d_{32}^0, d_{31}^0 + d_{12}^0) = \min(9, \infty + 10) = 9 \\
d_{34}^1 &= \min(d_{34}^0, d_{31}^0 + d_{14}^0) = \min(22, \infty + 5) = 22
\end{aligned}$$

in podobno. Dobimo

$$D^1 = \begin{bmatrix} 0 & 10 & 31 & 5 \\ 6 & 0 & 37 & 11 \\ \infty & 9 & 0 & 22 \\ 20 & 14 & 18 & 0 \end{bmatrix}$$

Ko v naslednjem koraku k povečamo iz 1 in 2 (najbolj zunanja for zanka v psevdokodu), dobimo

$$D^2 = \begin{bmatrix} 0 & 10 & 31 & 5 \\ 6 & 0 & 37 & 11 \\ 15 & 9 & 0 & 20 \\ 20 & 14 & 18 & 0 \end{bmatrix}$$

Zadnji dve matriki D^3 in D^4 (ta predstavlja rezultat algoritma!) napravite za vajo sami.

4 Naloge

1. Napišite program, ki poišče optimalno zasedenost nahrbtnika na dinamičen način.
2. Napišite program, ki poišče dolžine najkrajših poti na dinamičen način. V programu omogočite, da boste lahko graf prebrali tudi iz datoteke.

Za oceno 6: Obvezna je naloga 1.

Rok za oddajo obveznih nalog po elektronski pošti je: 31. 5. 2009 do 13:01.

Navodila za pošiljanje vaj najdete na spletu. Vaje pošljite v skladu z navodili na naslov: vaje.racunalnistvo@gmail.com.

Kopiranje rešitev se bo kaznovalo.