Algoritmi in podatkovne strukture 2

Dinamično programiranje (2. del)

Luka Fürst

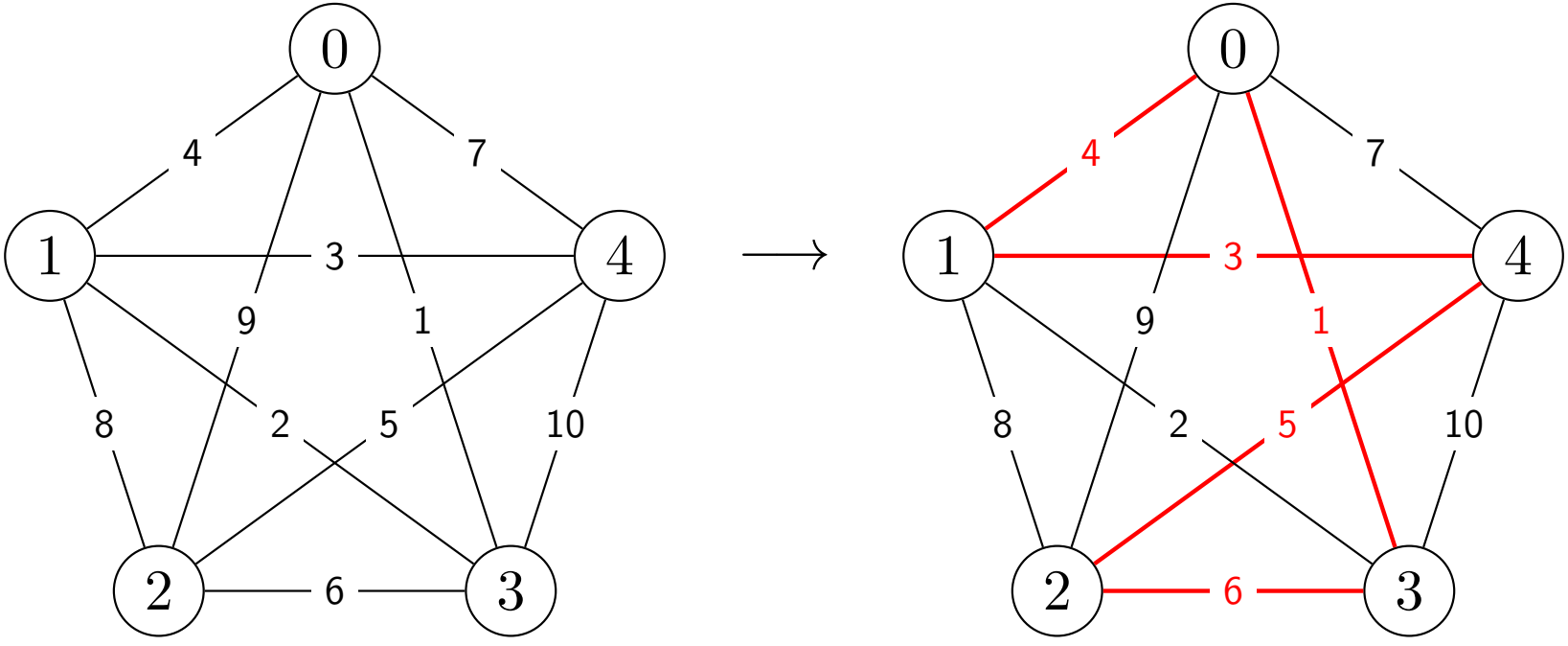
Problem trgovskega potnika

- Poišči najcenejši Hamiltonov cikel v polnem neusmerjenem grafu z utežmi (cenami) na povezavah
- Hamiltonov cikel je sprehod, ki se prične in zaključi v istem vozlišču, vsako od ostalih vozlišč pa obišče natanko enkrat

Formalneje . . .

- Podan je neusmerjen graf $G = (V, E)$ z $V = \{0, 1, \ldots, n-1\}$
- Naj bo $c(u, v)$ cena povezave (u, v)
- Naj bo $S(u_1 \to u_2 \to \ldots \to u_k)$ cena sprehoda $u_1 \to u_2 \to \ldots \to u_k$
- Minimiziraj $S(\sigma(0) \to \sigma(1) \to \ldots \to \sigma(n-1) \to \sigma(0))$ preko vseh možnih permutacij σ množice $\{0, \ldots, n-1\}$
- Fiksirajmo $\sigma(0) = 0$
  - cikel lahko vedno pričnemo in zaključimo v vozlišču 0

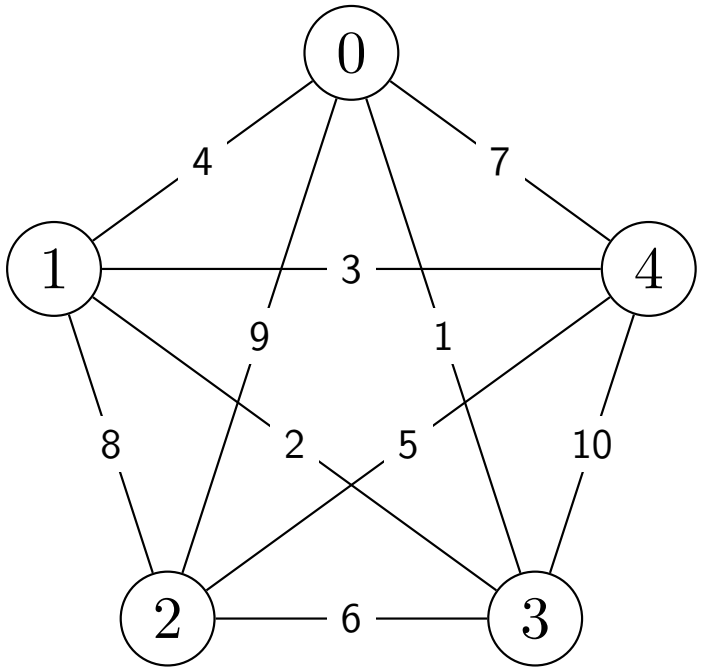
Primer



- Najcenejši Hamiltonov cikel: $0 \to 1 \to 4 \to 2 \to 3 \to 0$
- cena $= 19$

Naivno reševanje

- Preizkusimo vse možne permutacije vozlišč $1, 2, \ldots, n-1$
- $(n-1)!$ permutacij, $O(n)$ za vsako $\implies$ $O(n!)$



$$S(0 \to 1 \to 2 \to 3 \to 4 \to 0) = 35$$
$$S(0 \to 1 \to 2 \to 4 \to 3 \to 0) = 28$$
$$S(0 \to 1 \to 3 \to 2 \to 4 \to 0) = 24$$
$$\ldots$$
$$S(0 \to 1 \to 4 \to 2 \to 3 \to 0) = 19$$
$$\ldots$$
$$S(0 \to 4 \to 3 \to 2 \to 1 \to 0) = 35$$

Boljša ideja

- Optimalni cikel $0 \rightsquigarrow 0$ zgradimo tako, da
  - pričnemo s povezavo $0 \to k$ za nek $k \in \{1, \ldots, n-1\}$
  - poiščemo optimalno pot $k \rightsquigarrow 0$
- Izberemo k, ki vodi do minimalne skupne cene povezave $0 \to k$ in poti $k \rightsquigarrow 0$

Rekurenčna enačba

- Naj bo $u \in V$ in $M \subseteq V \setminus \{0, u\}$
- Naj bo $S(u \to M)$ cena optimalnega sprehoda, ki
  - se prične v vozlišču u
  - v nekem vrstnem redu obišče vsa vozlišča v M (vsako natanko enkrat)
  - se konča v vozlišču 0
- Iščemo $S(0 \to \{1, 2, \ldots, n-1\})$

Rekurenčna enačba

- Izračunajmo $S(u \to M)$
- $S(u \to \emptyset) = c(u, 0)$
  - od u gremo neposredno do 0
- Če je $M = \{v_1, \ldots, v_k\}$ za $k \geq 1$, potem

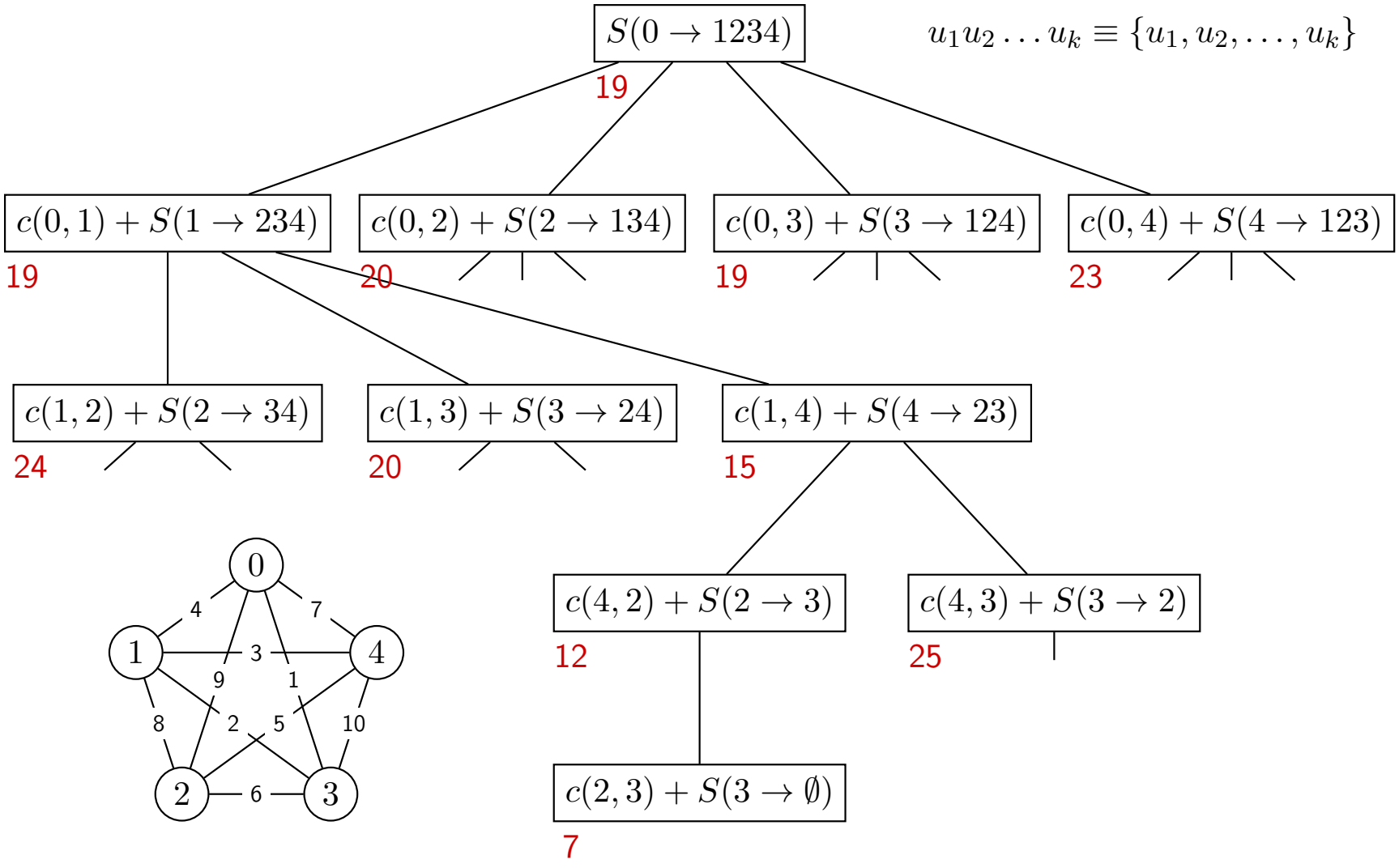
$$S(u \to M) = \min_{i=1}^{k}(c(u, v_i) + S(v_i \to M \setminus \{v_i\}))$$

  - najprej od u neposredno do nekega $v_i \in M$
  - potem od v_i prek ostalih vozlišč iz M do 0

Primer ($n = 5$)

$$
\begin{aligned}
&S(0 \to \{1,2,3,4\}) = \min\{ \\
&\qquad c(0,1) + S(1 \to \{2,3,4\}), \\
&\qquad c(0,2) + S(2 \to \{1,3,4\}), \\
&\qquad c(0,3) + S(3 \to \{1,2,4\}), \\
&\qquad c(0,4) + S(4 \to \{1,2,3\}) \\
&\} \\
&S(1 \to \{2,3,4\}) = \min\{ \\
&\qquad c(1,2) + S(2 \to \{3,4\}), \\
&\qquad c(1,3) + S(3 \to \{2,4\}), \\
&\qquad c(1,4) + S(4 \to \{2,3\}) \\
&\} \\
&\ldots
\end{aligned}
$$

Primer


$S(0 \to 1234)$
19
$u_1u_2\ldots u_k \equiv \{u_1, u_2, \ldots, u_k\}$
$c(0,1) + S(1 \to 234)$
19
$c(0,2) + S(2 \to 134)$
20
$c(0,3) + S(3 \to 124)$
19
$c(0,4) + S(4 \to 123)$
23
$c(1,2) + S(2 \to 34)$
24
$c(1,3) + S(3 \to 24)$
20
$c(1,4) + S(4 \to 23)$
15
$c(4,2) + S(2 \to 3)$
12
$c(4,3) + S(3 \to 2)$
25
$c(2,3) + S(3 \to \emptyset)$
7
0
1
2
3
4
4
7
3
9
1
8
2
5
10
6

Dinamično programiranje

- Če $S(u \to M)$ naivno računamo po rekurenčni enačbi, dobimo $O((n-1)!)$
- Lahko pa upoštevamo možnost, da se isti podproblem pojavi na več mestih v drevesu
  - npr. podproblem $S(1 \to \{3\})$ nastopa tako v izračunu $S(2 \to \{1,3,4\})$ kot v izračunu $S(4 \to \{1,2,3\})$
- Pa smo spet pri dobrem starem dinamičnem programiranju!
  - vsak podproblem $S(u \to M)$ izračunamo največ enkrat

Računanje $S(u \to M)$ z dinamičnim programiranjem

- $S(u \to M)$ računamo bodisi z memoizirano rekurzijo ali pa iterativno na podlagi manjših podproblemov
- V vsakem primeru potrebujemo dvodimenzionalno tabelo D
- Vrstični indeks je vozlišče u, stolpčni pa ... množica M?
- Kako lahko množico M pretvorimo v celoštevilski indeks?

Predstavitev množice M

- Množico M lahko predstavimo z njenim karakterističnim vektorjem
- Recimo, da vozlišču 1 ustreza skrajno levi, vozlišču n pa skrajno desni bit
  - Množici $\{1, 2, 4\}$ ustreza vektor 1101
- Stolpčni indeks v tabelo D je potemtakem zgolj desetiška predstavitev karakterističnega vektorja
  - Množici $\{1, 2, 4\}$ potemtakem pripada indeks 13

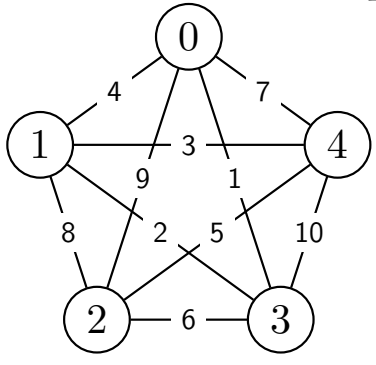
Tabela D

- Tabela D bo torej velika $n \times 2^{n-1}$
- Vrednost $S(u \to M)$ shranimo v celico $D[u, b(M)]$, kjer je $b(M)$ desetiška predstavitev karakterističnega vektorja podmnožice M
- Pri iterativnem polnjenju tabele D obravnavamo množice M po naraščajočih $b(M)$
  - če je $M_1 \subseteq M_2$, je $b(M_1) \leq b(M_2)$

Primer

smer polnjenja $\longrightarrow$

u																
0	0	14	2	18	18	21	16	19	8	14	7	13	21	21	19	19
1	4	10	3	14	17	17	15	15	4	10	3	9	17	17	15	15
2	9	12	7	16	9	12	7	16	12	12	11	11	12	12	11	11
3	1	17	1	17	15	18	13	18	6	12	5	12	18	18	17	17
4	7	7	11	11	14	14	12	12	7	7	6	6	17	17	16	16
	$\emptyset$	4	3	3 4	2	2 4	2 3	2 3 4	1	1 4	1 3	1 3 4	1 2	1 2 4	1 2 3	1 2 3 4



$\longrightarrow M \longrightarrow$

Sive vrednosti ne »obstajajo«,
a jih lahko brez škode izračunamo.

Rekonstrukcija najcenejšega Hamiltonovega cikla

- Vzdržujemo tabelo Z, ki je enako velika kot D
- V $Z[u, b(M)]$ shranimo vozlišče v, za katero velja

$$S(u \to M) = c(u, v) + S(v \to M \setminus \{v\})$$

- Naj bo $v_1 = Z[0,\ b(\{1, \ldots, n\})]$
- Začetek cikla je torej $0 \to v_1$
- Naj bo $v_2 = Z[v_1,\ b(\{1, \ldots, n\} \setminus \{v_1\})]$
- Cikel nadaljujemo z vozliščem v_2
- In tako naprej ...

Časovna zahtevnost

- Tabela D ima $n \times 2^{n-1} = O(2^n n)$ celic
- Za izračun vsake celice potrebujemo $O(n)$ časa
- Skupaj $O(2^n n^2)$

Spuščanje jajc

- Stavba ima N nadstropij
- Imamo K enakih jajc
- Radi bi s čim manj spusti jajc določili trdnost posameznega jajca (najvišje nadstropje, s katerega ga lahko spustimo, da se ne razbije)
- Koliko spustov potrebujemo v najslabšem primeru?
- Če jajce preživi padec z določenega nadstropja, preživi tudi padce z vseh nižjih nadstropij
- Če se jajce razbije pri spustu z nekega nadstropja, bi se razbilo tudi pri spustu z vseh višjih nadstropij

Spuščanje jajc

- Naj bo $S(n, k)$ minimalno število spustov pri n zaporednih nadstropjih in k jajcih
- Najprej obravnavajmo oba robna primera ...
- Koliko je $S(0, k)$?
- Koliko je $S(n, 1)$?

Robna primera

- $S(0, k) = 0$
  - če stavba nima nadstropij, nam ni treba ničesar spuščati
- $S(n, 1) = n$
  - če imamo eno samo jajce, nimamo druge možnosti, kot da ga po vrsti spuščamo s prvega, drugega, tretjega ... nadstropja
  - če ne bi delali tako, bi lahko »pravo« nadstropje izpustili
  - v najslabšem primeru torej potrebujemo n spustov

Dva konkretna primera

- $S(2,2) = 2$
  - v najslabšem primeru moramo preizkusiti obe nadstropji (dodatno jajce nam ne pomaga)
- Koliko je $S(3,2)$?

Koliko je $S(3,2)$?

- Možnost 1
  - prvo jajce spustimo z nadstropja 1
  - če se razbije, imamo odgovor $\implies$ $1+0=1$ spust
  - če preživi, nam ostaneta 2 jajci za 2 nadstropji $\implies$ $1+2=3$ spusti
  - v najslabšem primeru: 3 spusti
- Možnost 2
  - prvo jajce spustimo z nadstropja 2
  - če se razbije, nam ostane 1 jajce za 1 nadstropje $\implies$ $1+1=2$ spusta
  - če preživi, nam ostaneta 2 jajci za 1 nadstropje $\implies$ $1+1=2$ spusta
  - v najslabšem primeru: 2 spusta

Koliko je $S(3, 2)$?

- Možnost 3
  - prvo jajce spustimo z nadstropja 3
  - če se razbije, nam ostane še eno jajce za 2 nadstropji $\implies$ $1 + 2 = 3$ spusti
  - če preživi, imamo odgovor $\implies$ $1 + 0 = 1$ spust
  - v najslabšem primeru: 3 spusti
- Najboljša je možnost 2, ki v najslabšem primeru zahteva 2 spusta
- $S(3, 2) = 2$

Koliko je $S(n, k)$?

- Recimo, da prvo jajce spustimo z nadstropja i
- Če se razbije, nam ostane še $k - 1$ jajc za $i - 1$ nadstropij $\implies 1 + S(i - 1, k - 1)$ spustov
- Če preživi, nam ostane še k jajc za $n - i$ nadstropij $\implies$ $1 + S(n - i, k)$ spustov
- Upoštevati moramo slabši scenarij
- Število spustov je potemtakem $1 + \max(S(i - 1, k - 1), S(n - i, k))$
- Kako izberemo i?

Kako izberemo nadstropje za prvi spust?

- Tako, da dobimo minimalno število spustov

$$S(n,k) = \begin{cases} 0 & \text{pri } n = 0; \\ n & \text{pri } k = 1; \\ 1 + \min_{i=1}^{n}\{\max(S(i-1,k-1), S(n-i,k))\} & \text{sicer} \end{cases}$$

Algoritem

- Rekurenčne enačbe za $S(n, k)$ ni težko pretvoriti v rešitev z dinamičnim programiranjem
- Izračunamo $S(n, k)$ za vse pare $n \in \{0, \ldots, N\}$ in $k \in \{1, \ldots, K\}$
- Za izračun $S(n, k)$ poiščemo minimum prek n vrednosti
- Časovna zahtevnost: $O(N^2 K)$

Izboljšava

- Poglejmo na problem z druge strani
- Katero je pri podanem številu spustov S in podanem številu jajc K najvišje nadstropje N, da je $S(N, K) = S$?
- Označimo to število z $N(S, K)$
- Primer
  - $S(4, 2) = S(5, 2) = S(6, 2) = 3$
  - $S(7, 2) = 4$
  - Torej je $N(3, 2) = 6$
  - Če imamo na voljo 3 spuste in 2 jajci, potem lahko ima stavba največ 6 nadstropij, da lahko zanesljivo določimo trdnost jajca

Kako izračunamo $N(s, k)$?

- Prvo jajce spustimo z neznanega optimalnega nadstropja x
  - to ni najvišje nadstropje, pri katerem jajce preživi, pač pa nadstropje, ki bo pri optimalni strategiji spuščanja vodilo do največje vrednosti $N(s, k)$
- Če se jajce razbije, imamo na voljo še $s - 1$ spustov in $k - 1$ jajc, da ugotovimo, katero od $N(s - 1, k - 1)$ nadstropij pod nadstropjem x je domet jajca (najvišji spust, ki ga jajce preživi)
- Če jajce preživi, imamo na voljo še $s - 1$ spustov in k jajc, da ugotovimo, katero od $N(s - 1, k)$ nadstropij nad nadstropjem x je domet jajca
- Torej je $N(s, k) = 1 + N(s - 1, k - 1) + N(s - 1, k)$

Primer: izračun $N(3, 2)$

- 3 spusti, 2 jajci
- Prvo jajce je treba spustiti z nadstropja 3
- Če se razbije, imamo še 2 spusta in 1 jajce, da ugotovimo, katero od nadstropij 1 in 2 je domet jajca
- Če preživi, imamo še 2 spusta in 2 jajci, to pa zadošča za ugotavljanje trdnosti za 3 nadstropja, torej za nadstropja 4, 5 in 6
- $N(3, 2)$ je potemtakem $1 + N(2, 1) + N(2, 2) = 6$

$N(s,k)$ in prvotna uganka

$$N(s,k) = \begin{cases} 0 & \text{pri } s = 0; \\ s & \text{pri } k = 1; \\ 1 + N(s-1,k-1) + N(s-1,k) & \text{sicer} \end{cases}$$

- Kako lahko $N(s,k)$ uporabimo za rešitev prvotne uganke?
  - izračunaj $S(n,k)$, torej število spustov, potrebnih za določitev trdnosti pri n nadstropjih in k jajcih

$N(s,k)$ in $S(n,k)$

- $S(n,k)$ je najmanjši s, pri katerem je $N(s,k) \geq n$
- Primer
  - $N(3,2) = 6$ in $N(4,2) = 10$
  - zato je $S(6,2) = 3$ in
    $S(7,2) = S(8,2) = S(9,2) = S(10,2) = 4$
- $N(s,k)$ pri fiksnem k narašča z naraščajočim s
  - $N(2,2) = 3,\ N(3,2) = 6,\ N(4,2) = 10,\ \ldots$
- Zato lahko iskani s poiščemo z bisekcijo
  - zgornja meja za s je n

Učinkovitejši izračun $S(N, K)$

```
function S(n, k)
    ℓ ← 1
    d ← n
    while ℓ ≤ d do
        s ← (ℓ + d) / 2
        n' ← N(s, k)
        if n' = n then
            return s
        if n' > n then
            d ← s − 1
        else
            ℓ ← s + 1
    return ℓ
```

```
function N(s, k)
    if s = 0 then
        return 0
    if k = 1 then
        return s
    if D[s, k] = −1 then
        D[s, k] ← 1 + N(s − 1, k − 1) +
                     N(s − 1, k)
    return D[s, k]
```

- Tabelo D velikosti $(N + 1) \times (K + 1)$ inicializiramo z -1
- Funkcijo S pokličemo kot $\mathrm{S}(N, K)$

Učinkovitejši izračun $S(N, K)$

- $O(\log N)$ klicev funkcije N
- $O(NK)$ za izračun funkcije N z memoizirano rekurzijo (ali z iterativnim pristopom)
- Skupaj $O(NK \log N)$

Še učinkovitejša rešitev

- Enačba

$$N(s,k) = 1 + N(s-1,k-1) + N(s-1,k)$$

spominja na enačbo

$$\binom{p}{q} = \binom{p-1}{q-1} + \binom{p-1}{q}$$

- Trditev:

$$N(s,k) = \sum_{i=1}^{k} \binom{s}{i}$$

Še učinkovitejša rešitev

- Dokaz z indukcijo
- Primera $N(0,k)$ in $N(s,1)$ zlahka preverimo
- Predpostavimo veljavnost formule za $N(s-1,k-1)$ in $N(s-1,k)$

$$
\begin{aligned}
N(s,k) &= 1 + N(s-1,k-1) + N(s-1,k) \\
&= 1 + \sum_{i=1}^{k-1} \binom{s-1}{i} + \sum_{i=1}^{k} \binom{s-1}{i} \\
&= 1 + \binom{s-1}{1} + \left(\binom{s-1}{1} + \binom{s-1}{2}\right) + \left(\binom{s-1}{2} + \binom{s-1}{3}\right) \\
&\quad + \ldots + \left(\binom{s-1}{k-1} + \binom{s-1}{k}\right) \\
&= \binom{s}{1} + \binom{s}{2} + \ldots + \binom{s}{k} = \sum_{i=1}^{k} \binom{s}{i}
\end{aligned}
$$

Še učinkovitejša rešitev

- Iterativno računanje $N(s,k)$ za $k \in \{1,\ldots,K\}$

$$N(s,1) = \binom{s}{1}$$

$$N(s,2) = N(s,1) + \binom{s}{2}$$

$$N(s,3) = N(s,2) + \binom{s}{3}$$

$$\ldots$$

- $O(K)$ zadošča za izračun $N(s,k)$ za vse $k \in \{1,\ldots,K\}$
- Nabor vrednosti $N(s,.)$ izračunamo za $O(\log N)$ vrednosti s
- Skupaj $O(K \log N)$

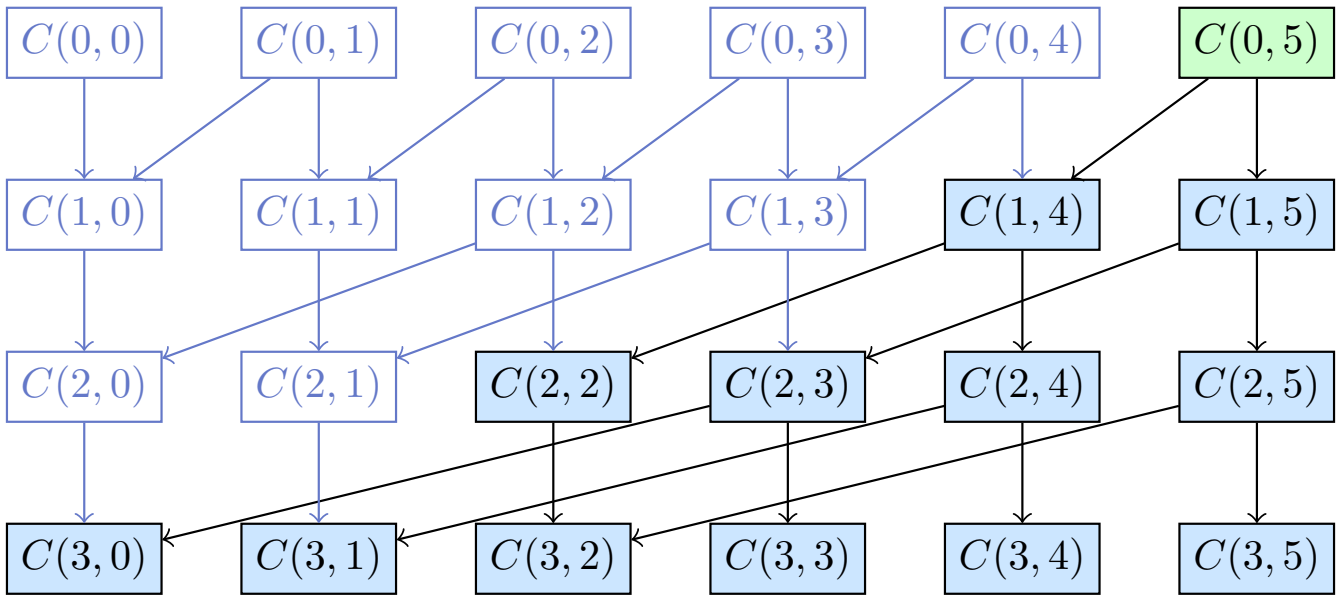
Dinamično programiranje kot graf

- Izvajanje algoritma po metodi dinamičnega programiranja si lahko predstavljamo kot usmerjen aciklični graf
- Primer: 0/1-nahrbtnik
- Vozlišča grafa (stanja)
  - $C(i, v)$ za različne $i \in \{0, \ldots, n\}$ in $v \in \{0, \ldots, V\}$
  - (maksimalna cena nahrbtnika prostornine v, v katerega lahko postavimo predmete i, $i+1$, $\ldots$, $n-1$)
- Povezave v grafu
  - $C(i, v) \to C(i+1, v)$
  - $C(i, v) \to C(i+1, v - v_i)$ (samo v primeru $v_i \leq v$)
- Funkcija prehodov

$$C(i,v) = \begin{cases} 0, & \text{če je } i = n; \\ C(i+1, v), & \text{če je } i < n \wedge v_i > v; \\ \max(C(i+1, v), c_i + C(i+1, v - v_i)), & \text{če je } i < n \wedge v_i \leq v \end{cases}$$

Graf za 0/1-nahrbtnik

- $V = 5$, $n = 3$
- prostornine predmetov: $\langle 1, 2, 3 \rangle$
- cene predmetov so lahko poljubne



Potovanje po grafu

- Od zgoraj navzdol
  - prični v začetnem vozlišču ($C(0, V)$)
  - obišči samo vozlišča, dosegljiva iz začetnega vozlišča
- Od spodaj navzgor
  - potuj v obratni smeri povezav od »listov« proti začetnemu vozlišču
  - izračunaj vrednosti za vsa vozlišča
- Prvi pristop opravi manj izračunov, a za ceno izvajanja rekurzije, manjše prijaznosti do predpomnilnika in večje prostorske zahtevnosti

Računska zahtevnost

- Prostor
  - $O(\text{število stanj})$
  - 0/1-nahrbtnik: $O(nV)$
  - Pri računanju od spodaj navzgor le $O(V)$, saj lahko graf obiskujemo po nivojih
- Čas
  - $O(\text{št. stanj} \times \text{zahtevnost izračuna prehoda med stanjema})$
  - 0/1-nahrbtnik: $O(nV \cdot 1) = O(nV)$

Pregled primerov

Problem	Stanja in prehod (splošni primer)	Čas
Nahrbtnik	$C(i,v) = \max(C(i+1,v),\ c_i + C(i+1, v-v_i))$	$O(nV \cdot 1)$
NNP	$\ell(i) = 1 + \max\{\ell(j) \mid j > i \wedge a_j > a_i\}$	$O(n \cdot n)$
Matrike	$M(i,j) = \min_{t=i}^{j-1}(M(i,t) + M(t+1,j) + v_i v_{t+1} v_{j+1})$	$O(n^2 \cdot n)$
Potnik	$S(u \to M) = \min_{i=1}^{k}(c(u,v_i) + S(v_i \to M \setminus \{v_i\}))$	$O(2^n n \cdot n)$
Jajca	$S(n,k) = 1 + \min_{i=1}^{n} \max(S(i-1,k-1), S(n-i,k))$	$O(NK \cdot N)$

V razmislek . . .

- Veliko problemov je mogoče definirati na rekurziven način
- Zakaj nekaterih od njih ni mogoče (ali ni smiselno) reševati z dinamičnim programiranjem?
- Najbolj znana primera takšnih problemov:
  - osem dam
  - skakačev obhod