Dynamic Programming

Oliver-Matis Lill

November 13, 2016

Sessiooni struktuur

- Sissejuhatus ja soojendus
- Bitmask DP
- Travelling Salesman probleem
- Kokkuvõte, üldised vihjed
- Harjutamine

Sissejuhatus

- Sarnane matemaatilisele induktsioonile
- Tüüpiline protseduur on järgmine
  1. Jaota probleem mingiks hulgaks alamprobleemideks
  2. Leia valem millega saab "suurema" alamprobleemi arvutada "väiksemate" alamprobleemide tulemuste abil
  3. Rakenda valemit et lahendada iga alamprobleem ja lõpuks ka esialgne probleem
- Dünaamiline planeerimise kasutamiseks läheb vaja alamprobleemideks jaotamise skeemi ja rekursioonivalemit mis kirjeldab alamprobleemide vahelisi seoseid
- Samal probleemil võib olla mitu skeemi ja rekursioonivalemit.
- Kui skeem loob $O(S)$ alamprobleemi ja rekursioonivalemi arvutamiseks tuleb teha $O(R)$ tehet, siis keerukus tuleb $O(SR)$

Ülesanne "Vacations"

- n puhkuspäeva, igal puhkuspäeval võib olla spordisaal avatud ja/või toimuda programmeerimisvõistlus
- Vasya ei ole nõus tegema sama asja kaks päeva järjest
- Loo puhkuse jaoks vasjale plaan millel on minimaalne hulk päevi kus Vasya ei võistle ega tee sporti
- Sobiv aga ebaoptimaalne plaan näitel 2:

Päev:	1	2	3	4	5	6	7
Üritused:	v	v,s	v,s	s	v	s	v,s
Võistleb:	jah	ei	jah	ei	jah	ei	jah
Treenib:	ei	jah	ei	ei	ei	jah	ei

Seal on üks puhkepäev, nimelt päev 4

Ülesanne "Vacations"

- Paneme tähele, et igal uuel päeval meid huvitab ainult see mida me eelneval päeval tegime
- Seega alamprobleemid võiksid vastata järgmisele skeemile:
  $D_i^{(c)}$ = minimaalne pukepäevade arv millega saab veeta esimesed i puhkepäeva kus viimase puhkepäeva tegevust kirjeldab c

  Siin$c = 0$ tähendab puhkust, $c = "v"$ tähendab võistlust ja $c = "s"$ tähendab sporti
- Rekursioonivalemid tulevad:

$$D_{i+1}^{(0)} = \min(D_i^{(0)}, D_i^{(v)}, D_i^{(s)}) + 1$$
$$D_{i+1}^{(v)} = \min(D_i^{(0)}, D_i^{(s)})$$
$$D_{i+1}^{(s)} = \min(D_i^{(0)}, D_i^{(v)})$$

- Vastus on: $\min(D_n^{(0)}, D_n^{(v)}, D_n^{(s)})$

Bitmask-i skeem

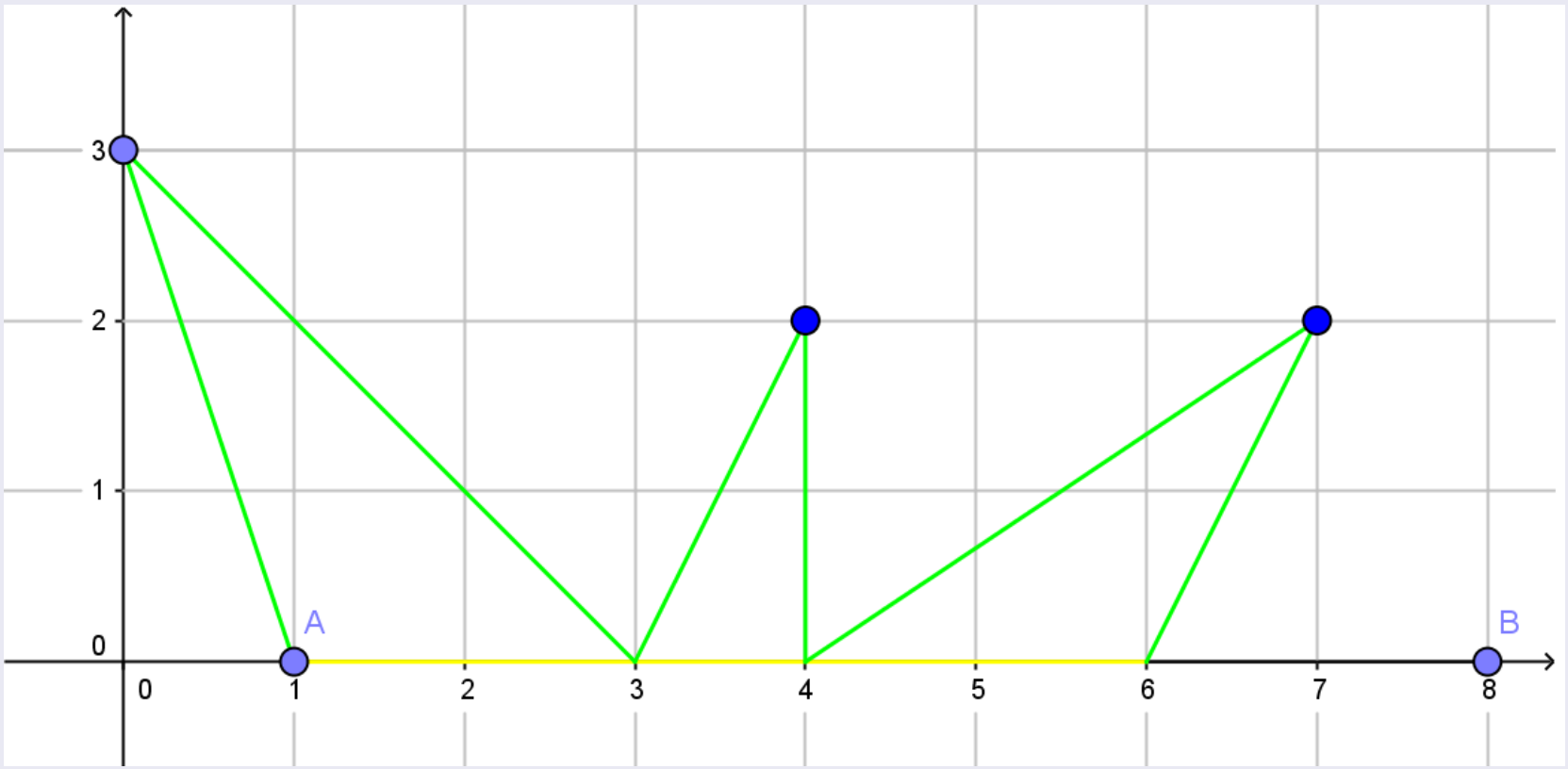
- Idee on, et kui probleem nõuab mingi hulga peal mingi väärtuse arvutamist, siis alamprobleemideks jaotamine toimub alamhulkade järgi
- Siin saab nn "bitmask"-e kasutada alamhulkade tähistamisel. Hulgal $S = \{x_1, x_2, x_3, x_4, x_5\}$ bitmask 01101 tähistab alamhulka $S_1 = \{x_2, x_3, x_5\}$
- Tüüpiliselt taandab algoritmi keerukuse $O(n!)$ pealt $O(2^n)$ või mingi sarnase keerukuse peale.

Ülesanne "Bear and Floodlight"

- n tulvarit tasandil, neid saab keerata ja igaüks valgustab mingit nurka a_i
- Karu soovib liikuda sirgjooneliselt punktist $(l, 0)$ punkti $(r, 0)$ mööda valgustatud teed.
- Suuna tulvarid nii, et karu saaks valgustatud teed mööda võimalikult lähedale oma sihtkohale kõndida.

Ülesanne "Bear and Floodlight"

Näide



Ülesanne "Bear and Floodlight"

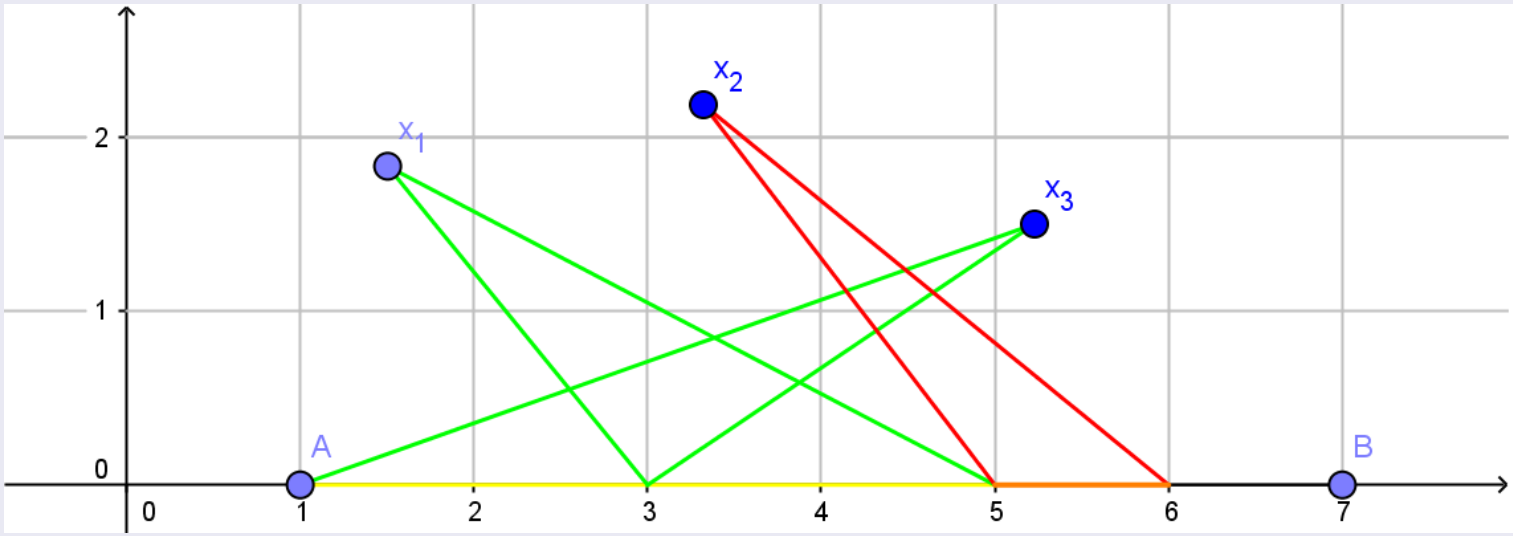
- Idee: Leia iga tulvarite alamhulga jaoks, kui kaugele on nendega võimalik kõndida
- Skeem:
  D_S = maksimaalne kaugus kuhu tulvarite hulgaga S on võimalik kõndida.
- Rekursioonivalem:

$$D_S = \max\{\mathrm{ext}(D_{S\setminus\{x\}}, x) | x \in S\}$$

Kus $\mathrm{ext}(d, x)$ on saadud valgustatud teekonna pikkus, kui me valgustame pikkusega d valgustatud piirkonda tulariga x edasi. On lihtne näha et x peaks olema suunatud nii, et selle valgustatud lõigu vasak serv oleks $(l + d, 0)$.

Ülesanne "Bear and Floodlight"

- $D_{\{x_1,x_2,x_3\}} = \max(\text{ext}(D_{\{x_1,x_2\}}, x_3), \text{ext}(D_{\{x_1,x_3\}}, x_2), \text{ext}(D_{\{x_2,x_3\}}, x_1))$
- Näide $\text{ext}(D_{\{x_1,x_3\}}, x_2)$ vastavale valikule:



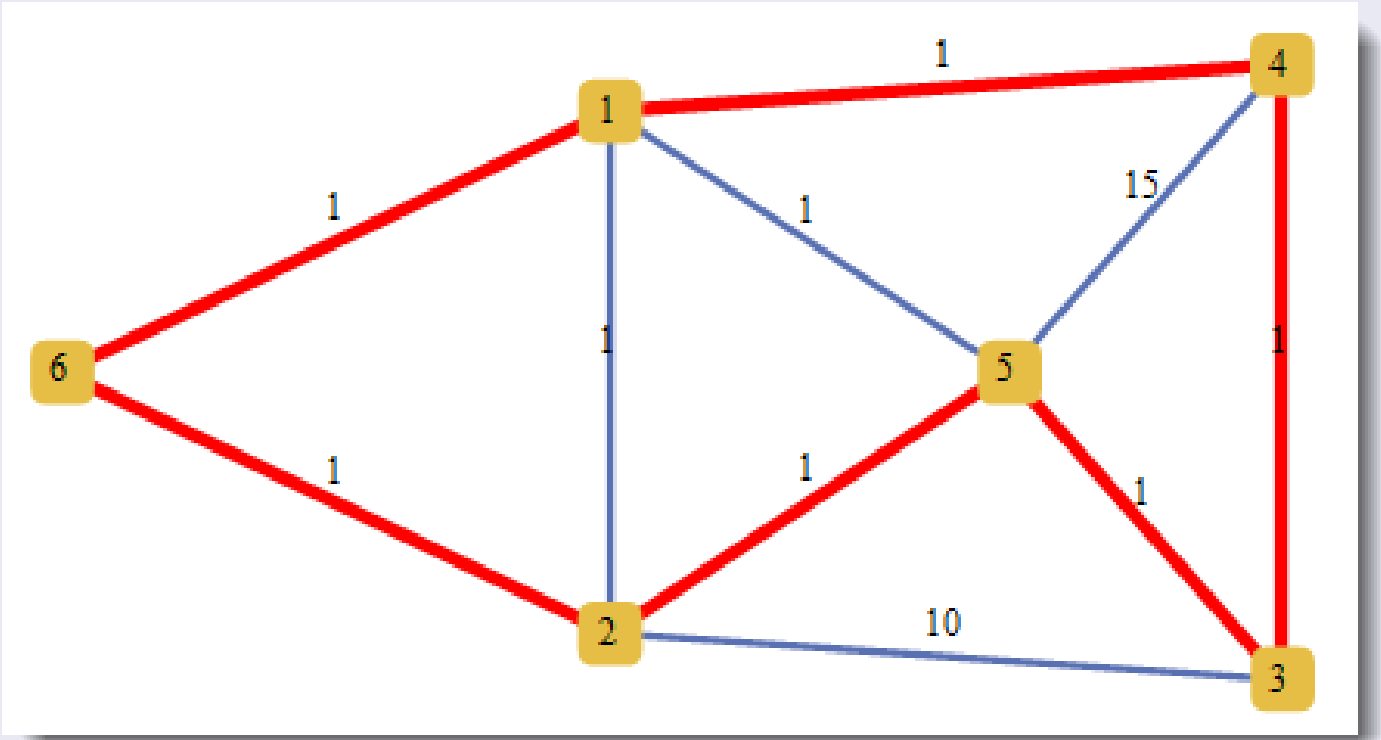
Ülesanne "Bear and Floodlight"

Tõestus

- On lihtne näha, et rekursiooni valem vaatab läbi korrektsed tularite suunamised ja arvutab nende suunamiste väärtused õieti.
- Vaatame mingit optimaalset tularite suunamise strateegiat tularite hulgale S. Vaatame seal kõige parempoolsema valgustuslõigu paremservaga tularit x.
- Paneme tähele et kui tularite hulga $S \setminus \{x\}$ suunata nii et nad valgustavad maksimaalse lõigu, siis vähemalt sama pikk lõik on valgustatud.
- Seega $\mathrm{ext}(D_{S\setminus\{x\}}, x)$ annab vähemalt sama hea lahendi kui praegune optimaalne lahendus, ehk on ise ka optimaalne.
- Seega rekursioonivalem tuvastab alamhulga jaoks läbi vähemalt ühe optimaalse suunamise.

Travelling Salesman probleem

- Meil on n linna ja linnade vahel on teatud pikkustega teed.
- Leia lühim ringkäik mis läbib köik linnad täpselt üks kord ja jõuab alguslinna tagasi.
- Näide Rändkaupmehe ringkäigust:



Travelling Salesman probleem

- Idee: See kuidas me edasi liikuda saame sõltub ainult sellest mis linnad me oleme varem läbinud ja kus linnas me praegu oleme.
- Skeem:
  $D_S^{(i)}$ =minimaalse teekonna pikkus millega on võimalik läbida hulgas S olevad linnad ja jõuda lõpuks linna i
- Rekursioonivalem:

$$D_S^{(i)} = \min\{D_{S\setminus\{i\}}^{(j)} + d_{j,i} | j \in S \setminus \{i\}\}$$

  Kus $d_{j,i}$ on tee pikkus linnast j linna i.
- Meil on $O(2^n)$ seisundit ja rekursioonivalemi saab arvutada keerukusega $O(n^2)$, seega meil on $O(n^2 2^n)$ algoritm.

Kokkuvõte ja soovitused

- DP on tõenäoliselt kõige levinum probleemilahendamise meetod võistlusprogrammeerimise valdkonnas üldse
- Tihti osa suuremast ülesannetest
- Raskemate ülesannete puhul võib rekursioonivalem väga keerukaks minna
- Probleemil võib olla mitu erinevat rekursioonivalemit ja alamprobleemide skeemi, mõned neist paremad kui teised.
- Sobiva reukrsioonivalemi ja skeemi tuvastamine võib raske olla
- Harjutamine teeb meistriks