Twaalfde college algoritmiek

3 mei 2022

Branch & Bound

- beoordelingen opdracht 1. . .
- opdracht 3. . .

Backtracking

- bouwt een oplossing component voor component op
- kijkt tijdens de stap-voor-stap constructie of de deeloplossing die bekeken wordt nog aan de gestelde restricties/eisen voldoet (kan voldoen)
- zo niet, breidt dan de deeloplossing niet verder uit en herziet de laatste keuze (backtrack!)
- werking te beschrijven m.b.v. een state space tree
- houdt (alleen) het pad corresponderend met de (deel)oplossing die 'nu' wordt uitgebreid bij
- spaart soms veel werk uit vergeleken met exhaustive search
- toepasbaar op allerlei problemen waarbij oplossingen stap voor stap gegenereerd kunnen worden

Optimalisatieprobleem: er wordt een oplossing gezocht met minimale of maximale ..... (b.v. bepaalMinSchema)

Terminologie:

- De functie/waarde die ge-optimaliseerd moet worden heet wel de objectfunctie (bijvoorbeeld de lengte van een Hamiltonkring)
- Een oplossing die voldoet aan de restricties van het probleem heet feasible (toelaatbaar)
- Een optimale oplossing is een/de toelaatbare oplossing met de beste waarde van de objectfunctie

Backtracking en optimalisatieproblemen

- Bij een minimalisatieprobleem kun je bijv. ook stoppen met uitbreiden wanneer je deeloplossing al een grotere waarde van de te optimaliseren functie heeft dan de huidige beste oplossing (die wordt dus bijgehouden/opgeslagen) en alleen maar nog groter kan worden.

- Als je snel een (bijna) optimale oplossing vindt, kunnen verdere deeloplossingen eerder worden afgekapt en ben je dus sneller klaar $\Longrightarrow$ gretig algoritme.

- Backtracking is het efficiëntst wanneer *een* oplossing gezocht wordt, dus voor optimalisatieproblemen is backtracking niet bij voorbaat de meest geschikte oplossingsmethode.

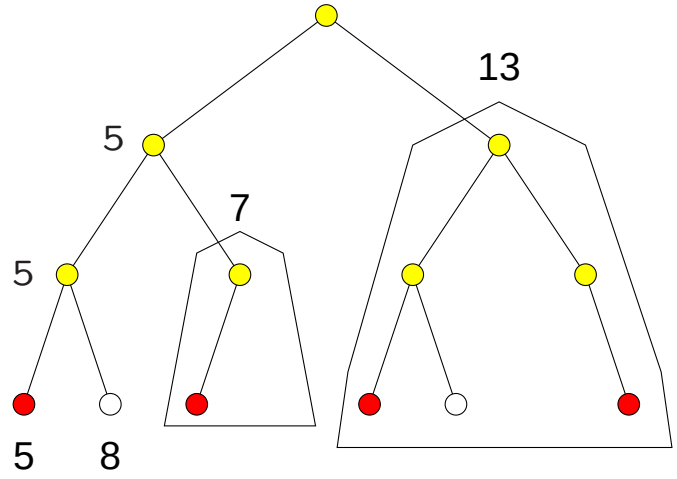
Assignmentprobleem (toewijzingsprobleem)

Gegeven n personen en n taken (jobs). Persoon i kan taak j doen voor `kosten`$[i][j]$ euro. **Gevraagd**: de/een toewijzing van de personen aan de jobs (één persoon per job en één job per persoon) met minimale kosten.

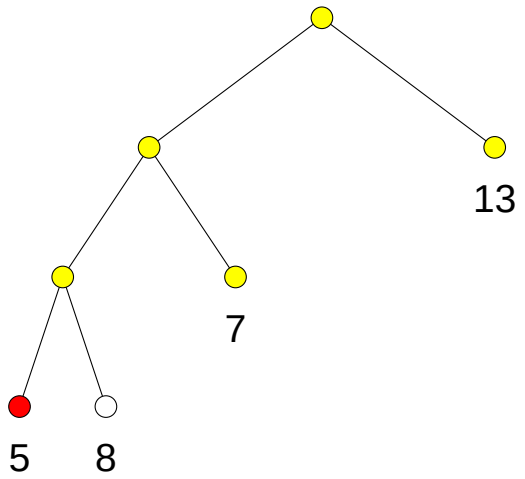
Voorbeeld:

	W	X	Y	Z
Alice	9	2	7	8
Bob	6	4	3	7
Carol	5	8	1	8
David	7	6	9	4

Gretig algoritme, en dan backtracking met afkappen. . .



state space tree



gesnoeide boom

Bij branch & bound berekenen we voor elke deeloplossing een ondergrens (of bovengrens).

Witte knopen corresponderen met toelaatbare oplossingen; rode knopen met niet-toelaatbare oplossingen; de waarden bij de bladeren geven de waarde van de objectfunctie van de bijbehorende oplossing; de waarden bij de andere knopen geven een ondergrens op de te verwachten waarde van de objectfunctie; bij de knoop met grens 13 kan met-een gesnoeid worden, die met 7 wordt nog bekeken, maar leidt tot een niet-toelaatbare oplossing

	W	X	Y	Z
Alice	9	2	7	8
Bob	6	4	3	7
Carol	5	8	1	8
David	7	6	9	4

Een ondergrens voor de kosten van een optimale oplossing. . .

	W	X	Y	Z
Alice	9	2	7	8
Bob	6	4	3	7
Carol	5	8	1	8
David	7	6	9	4

Een ondergrens voor de kosten van een optimale oplossing:

(a) neem uit elke rij de kleinste waarde en tel die bij elkaar op: $2 + 3 + 1 + 4 = 10$

(b) neem uit elke kolom de kleinste waarde en tel die bij elkaar op: $5 + 2 + 1 + 4 = 12$

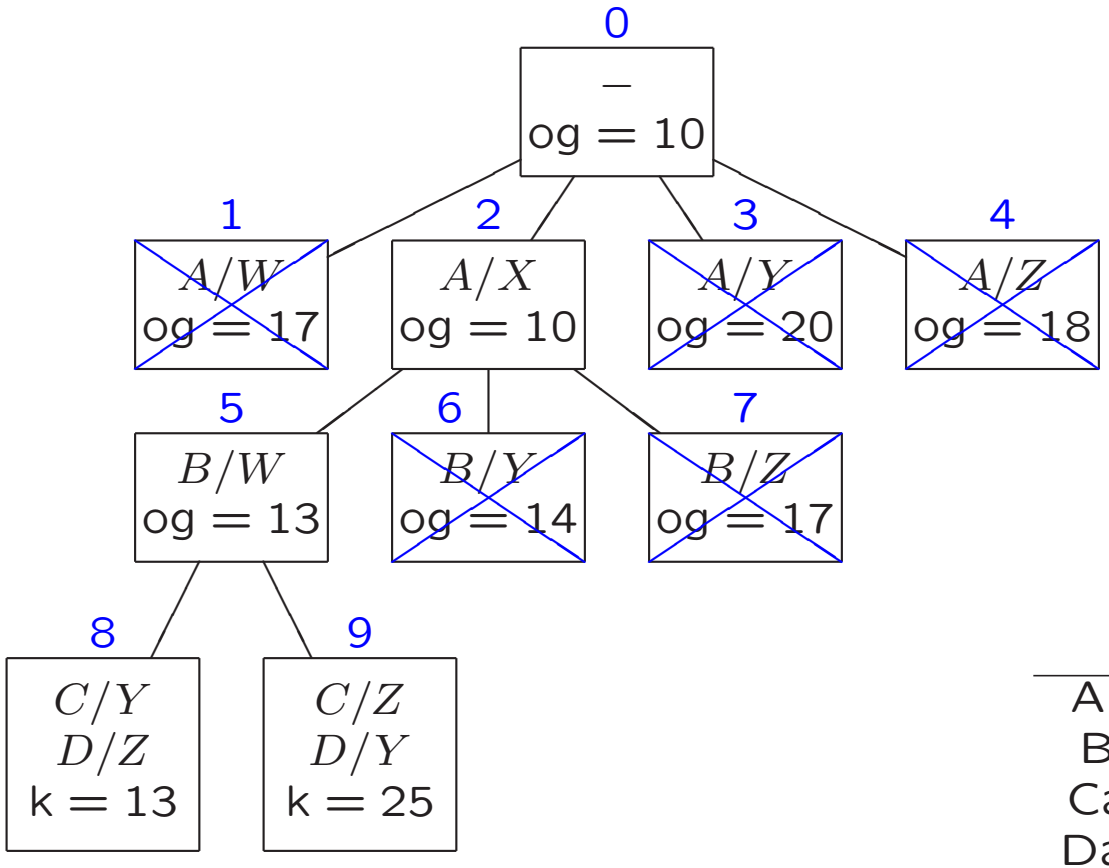
Betekenis ondergrens. . .

We genereren oplossingen door de personen een voor een aan een job te koppelen, en gebruiken daarbij de ondergrens volgens suggestie (a).
De optimale oplossing heeft totale kosten 13:
Alice job X; Bob job W; Carol job Y; David job Z

Voor een willekeurige knoop/deeloplossing berekenen we de ondergrens op de te verwachten waarde van de objectfunctie door uit elke verdere rij de kleinste waarde van de nog beschikbare jobs te nemen en deze op te tellen bij de waarde van de deeloplossing. Voor de deeloplossing(en) die Alice aan job W koppelt zal die ondergrens bijvoorbeeld $9+3+1+4 = 17$ zijn. (Analoog voor kolommen: kies uit elke verdere kolom de kleinste waarde van de nog beschikbare personen(*).)

De volgorde waarin de knopen van de state space tree worden uitgebreid laten we afhangen van de berekende ondergrens. We kiezen de knopen met de beste (=laagste) ondergrens als eerste: dit lijkt de meest veelbelovende knoop. Deze strategie heet wel de best-first branch-and-bound.

Priority queue nodig.



	W	X	Y	Z
Alice	9	2	7	8
Bob	6	4	3	7
Carol	5	8	1	8
David	7	6	9	4

Opgave: los het probleem op met de ondergrenzen berekend volgens (*).

Branch & bound

- is alleen toepasbaar op optimalisatieproblemen
- genereert oplossingen stap voor stap en houdt de tot dusver gevonden beste oplossing bij
- gebruikt voor elke deeloplossing (= knoop in de state space tree) een of andere ondergrens (minimalisatieprobleem) resp. bovengrens (maximalisatieprobleem) op de te verwachten waarde van de object-functie, met als doel
  - van deeloplossingen te kunnen bepalen dat ze niet verder bekeken hoeven te worden: snoeien
  - de zoekvolgorde in de zoekruimte (state space tree), dus de volg-orde waarin knopen worden gegenereerd en verder bekeken, te lei-den

Een branch-and-bound algoritme breidt een knoop (deeloplossing) niet verder uit als

- de waarde van de ondergrens (bovengrens) bij die knoop niet beter is dan de waarde van de tot dusver gevonden beste oplossing:
  als ondergrens $\geq$ tot dusver gevonden minimale waarde, dan snoeien

- de deeloplossing niet meer voldoet aan de restricties (of niet meer uit te breiden is tot een toelaatbare oplossing)

- er nog maar één volledige, toelaatbare oplossing mogelijk is bij de deeloplossing (i.h.b. als deeloplossing zo'n volledige oplossing *is*); de waarde van deze ene oplossing wordt met beste oplossing tot nu toe vergeleken; update zonodig beste oplossing.

De volgorde waarin de knopen (deeloplossingen) worden uitgebreid hangt direct af van de berekende grenzen:

- er worden meerdere deeloplossingen tegelijk bijgehouden, dit in tegenstelling tot backtracking
- in elke stap wordt een van al deze deeloplossingen gekozen, en daarvan worden alle 1-staps-uitbreidingen (kinderen in de state space tree) bekeken en geëvalueerd (d.w.z. ondergrens/bovengrens bepaald)
- zinloze uitbreidingen worden meteen verworpen
- meestal wordt de deeloplossing gekozen die het meest veelbelovend lijkt: best-first branch-and-bound
- bij minimalisatieproblemen (resp. maximalisatieproblemen) kiezen we de knoop met de laagste ondergrens (resp. hoogste bovengrens) als eerste

Werkcollege: Los het toewijzingsprobleem op voor onderstaand voorbeeld met behulp van

1. backtracking (snoeien op de kosten van deeloplossingen)
2. branch-and-bound

en vergelijk de hoeveelheid snoeiwerk bij beide methoden, alsmede de volgorde waarin de knopen van de state space tree worden bekeken.

	W	X	Y	Z
Alice	4	7	3	5
Bob	6	2	9	1
Carol	3	9	5	3
David	1	1	1	8

Voor minimalisatieprobleem, zoals toewijzingsprobleem,

met priority queue q:

```
q.insert (lege deeloplossing, met ondergrens);
best = maxint;

while (!q.isempty() && q.findmin().ondergrens < best) do
        huidig = q.deletemin();
        for alle 1-staps-uitbreidingen kind van huidig do
            if deeloplossing kind is toelaatbaar then
                if deeloplossing kind is volledige oplossing ||
                   er is maar 1 volledige oplossing mogelijk bij deeloplossing kind then
                    K = kosten van volledige oplossing;
                    if K < best then
                        onthoud volledige oplossing;
                        best = K;
                    fi
                else   // geen volledige oplossing
                    q.insert (kind, met ondergrens)
                fi
            fi
        od
od
```

Gegeven n objecten, met gewicht $w_1, \ldots, w_n$ en waarde $v_1, \ldots, v_n$, en een knapzak met capaciteit W.
Gevraagd: de meest waardevolle deelverzameling der objecten die in de knapzak past (dus met totaalgewicht $\leq W$).

Voorbeeld:

item	w	v	v/w
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Maximaal gewicht $W = 10$

item	w	v	v/w
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4
	$W = 10$		

Opbouwen van de oplossing: in de i-de stap wordt object i wel of niet gekozen,

Een bovengrens voor de waarde van een optimale oplossing:

- Bij aanvang: . . .
- Na de i-de stap: . . .

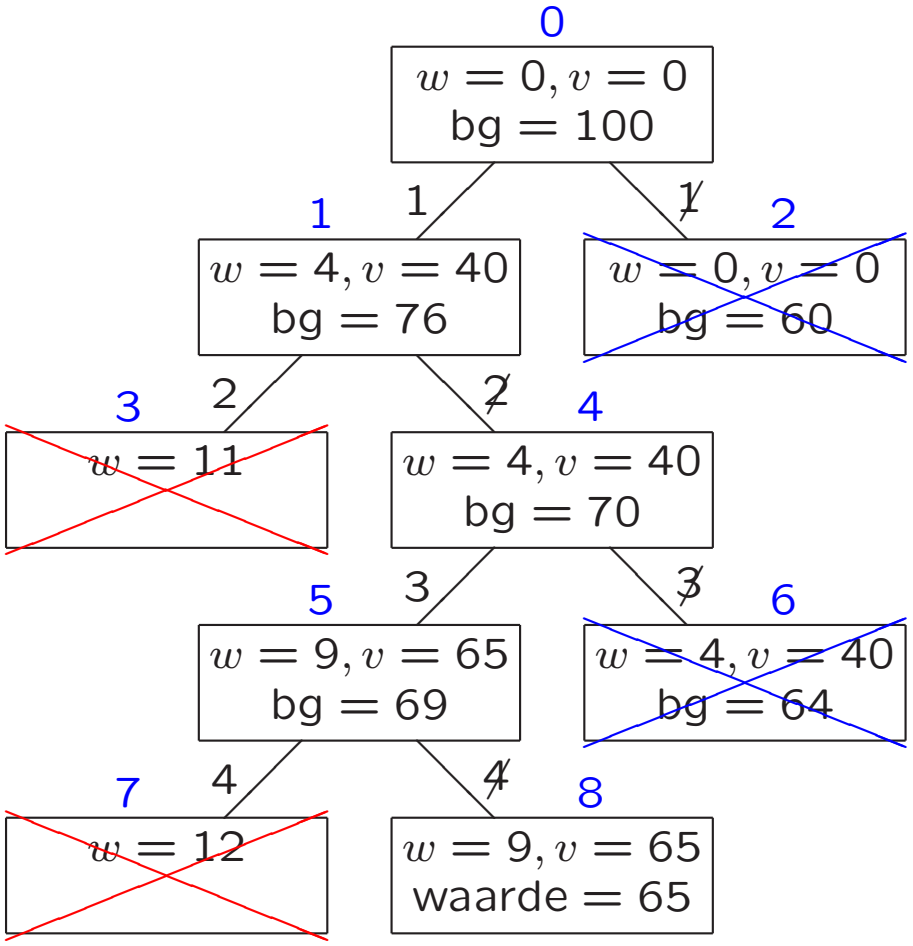
item	w	v	v/w
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4
$W = 10$			

Opbouwen van de oplossing: in de i-de stap wordt object i wel of niet gekozen,

Een bovengrens voor de waarde van een optimale oplossing:

- Bij aanvang: $W * (v_1/w_1) = 100$;
- Na de i-de stap: $v + (W - w) * (v_{i+1}/w_{i+1})$, met v de totaalwaarde van de reeds gekozen objecten en w het totaalgewicht daarvan.

De optimale oplossing heeft gewicht 9 en waarde 65: $\{1, 3\}$.



item	w	v	v/w
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

$W = 10$

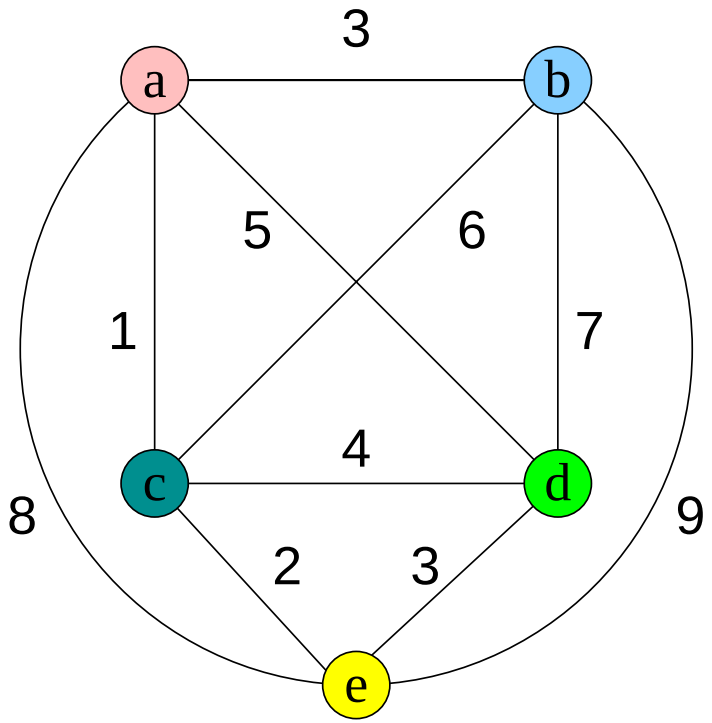
Zie ook exercise 12.2.5, Levitin.

Traveling Salesman Problem (handelsreizigersprobleem)

Gegeven n steden waarvan alle onderlinge afstanden bekend zijn.

Gevraagd: de/een kortste route die elke stad precies één keer aandoet, en weer terugkeert in het vertrekpunt.

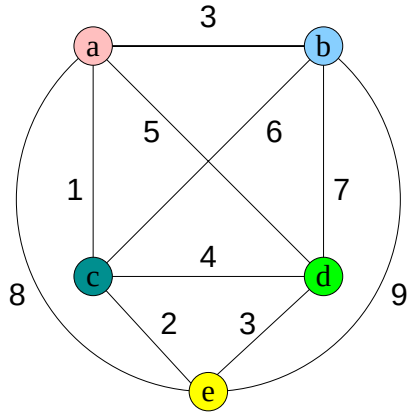
Ofwel: vind de/een kortste Hamiltonkring in een samenhangende gewogen (complete) graaf. Het gaat hier om een ongerichte graaf.



De optimale oplossing heeft lengte 16: a, b, d, e, c, a

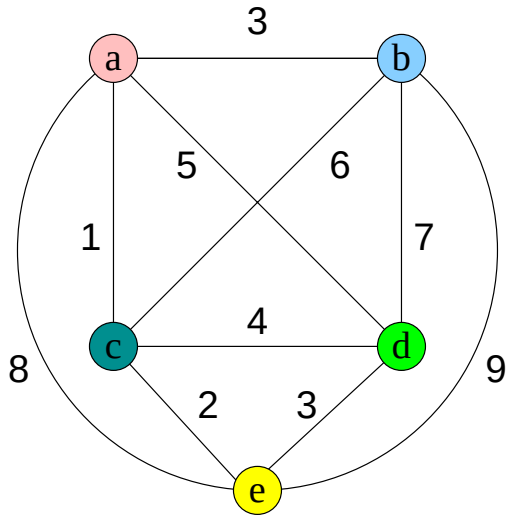
Mogelijke ondergrenzen voor de kosten van een optimale oplossing. . .

Mogelijke ondergrenzen voor de kosten van een optimale oplossing:



- eenvoudig: $n\times$ kortste afstand tussen twee knopen; $5 \times 1 = 5$ bij aanvang (analoog als reeds takken gekozen zijn).
- iets beter: de lengtes van de n kortste takken gesommeerd; $1 + 2 + 3 + 3 + 4 = 13$ bij aanvang (analoog als reeds takken gekozen zijn).
- nog beter: som over alle knopen i van de afstanden van knoop i tot de twee dichtstbijzijnde knopen (inclusief al gekozen takken), en dat gedeeld door 2(*); $((1+3)+(3+6)+(1+2)+(3+4)+(2+3))/2 = 14$ bij aanvang.
  Opmerking: delen door 2 is niet per se nodig; je kunt ook 2×lengte minimaliseren (*).

Goede/slechte ondergrens. . .



Ondergrens: som over alle knopen i van de afstanden van knoop i tot de twee dichtstbijzijnde knopen
(inclusief al gekozen takken), en dat gedeeld door 2;
$((1+3)+(3+6)+(1+2)+(3+4)+(2+3))/2 = 14$ bij aanvang.

State space tree. . .

De volgende versnellingen zijn mogelijk:

- bekijk alleen Hamiltonkringen startend in a
  immers, de kring c, d, e, a, b, c is hetzelfde als de kring a, b, c, d, e, a (en als d, e, a, b, c, d en e, a, b, c, d, e en als b, c, d, e, a, b).

- bekijk alleen situaties waarbij b wordt bezocht voor c
  de kring a, c, d, b, e, a is dezelfde kring als a, e, b, d, c, a maar in omgekerde volgorde. Beide hebben dezelfde lengte (want deze graaf is ongericht).

Gezien in state-space-tree: deeloplossingen beginnend met a, c zijn *ontoelaatbaar* (infeasible) na uitbreiding komt c voor b.
Deze knoop hoeft dus ook niet verder bekeken te worden; bijbehorende oplossingen komen we elders wel tegen. Bijv.: a, c, e, b, d, a vinden we terug als a, d, b, e, c, a, dus via knoop a, d.

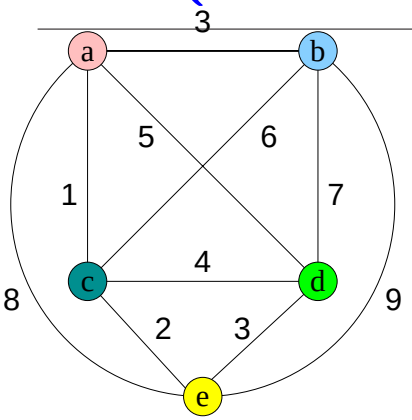
We zeiden wel:

> delen door 2 is niet per se nodig; je kunt ook 2×lengte minimaliseren (*)

Delen door 2 geeft echter wel scherpere ondergrens, als je resultaat afrondt naar boven.

In vorige voorbeeld werd ondergrens 31 van knoop a,d hiermee 16. Deze knoop hoefden we nu niet uit te werken.

Branch & Bound, state space tree (zonder door 2 delen)



0
a
og = 4+9+3
+7+5 = 28

1
a,b
og = 4+9+3
+7+5 = 28

2
a,c

3
a,d
og = 6+9+3
+8+5 = 31

4
a,e
og = 9+9+3
+7+10 = 38

5
a,b,c
og = 4+9+7
+7+5 = 32

6
a,b,d
og = 4+10+3
+10+5 = 32

7
a,b,e
og = 4+12+3
+7+11 = 37

8
a,d,b
og = 6+10+3
+12+5 = 36

9
a,d,c

10
a,d,e
og = 6+9+3
+8+5 = 31

13
a,b,c,d
(e,a)
2L = 11+9+10
+7+11 = 48

14
a,b,c,e
(d,a)
2L = 8+9+8
+8+5 = 38

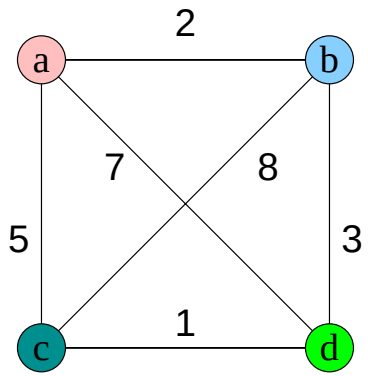
15
a,b,d,c
(e,a)
2L = 11+10+6
+11+10 = 48

16
a,b,d,e
(c,a)
2L = 4+10+3
+10+5 = 32

11
a,d,e,b
(c,a)
2L = 6+15+7
+8+12 = 48

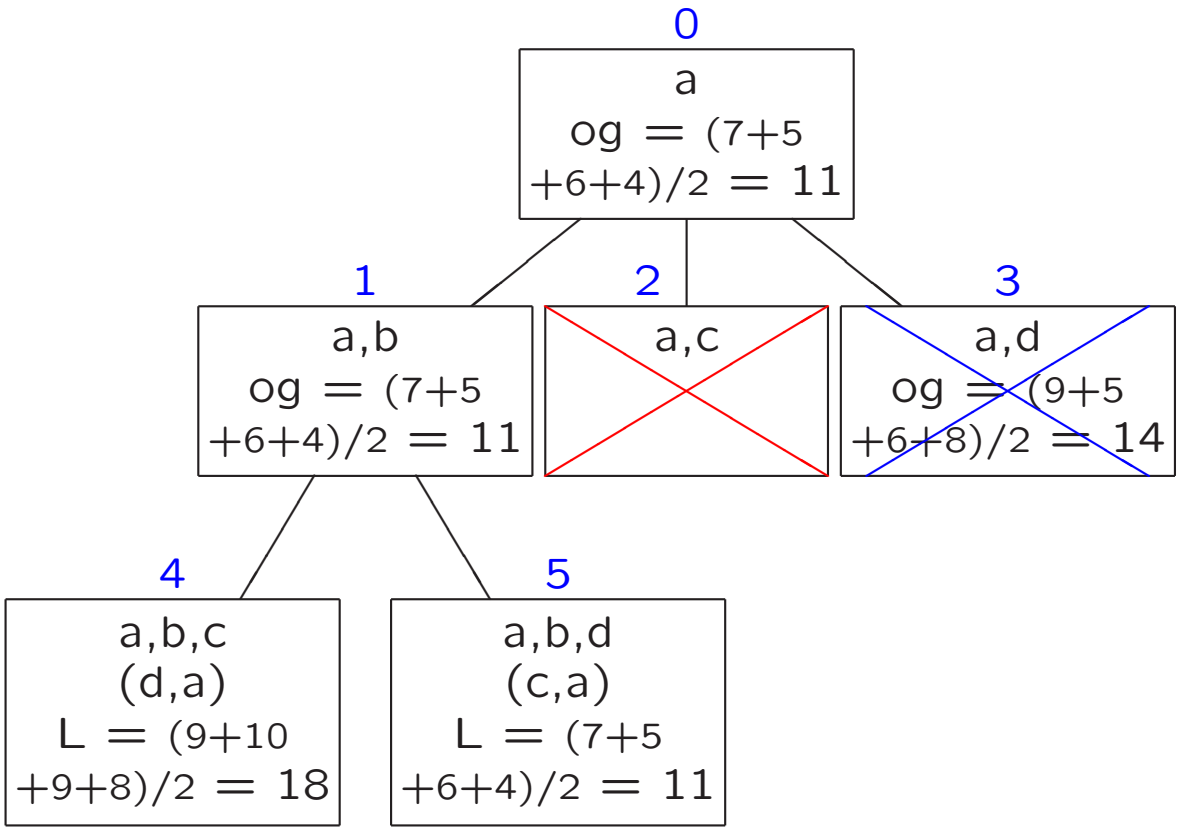
12
a,d,e,c

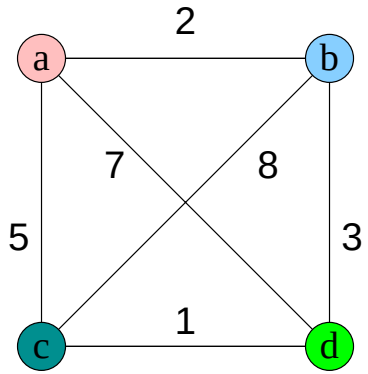
Werkcollege: Nog een voorbeeld, met als ondergrens wederom de som van de twee kortste takken per knoop (eventueel gedeeld door 2, zoals in het boek).



De optimale oplossing heeft lengte 11: a, b, d, c, a.

Exercise 12.2.9, vervolg





Gretig algoritme voor TSP

- geeft geen ondergrens (maar eerder een bovengrens),
- kan wel gebruikt worden om een deeloplossing eerder te verwerpen,
- kan nauwelijks gebruikt worden om meer deeloplossingen te verwerpen.

Overeenkomsten en verschillen tussen:

- Exhaustive search (ES)
- Backtracking (Bt)
- Best-first branch-and-bound (B&B)

N.B.: Verschillende punten hieronder hangen soms met elkaar samen.

- **ES** geschikt voor optimalisatieproblemen en andere problemen
  Bt geschikt voor optimalisatieproblemen en andere problemen
  B&B alleen geschikt voor optimalisatieproblemen

- **ES** kán oplossing component voor component opbouwen, maar is ook geschikt voor problemen waar geen sprake is van componenten en deeloplossingen

  Bt bouwt oplossing component voor component op

  B&B bouwt oplossing component voor component op

- **ES** controleert pas op geldigheid als het complete oplossing heeft opgebouwd (in geval dat ES oplossing component voor component opbouwt, ook verderop)

  Bt breidt deeloplossing niet verder uit als het geen geldige complete oplossing meer kan worden

  B&B breidt deeloplossing niet verder uit als het geen geldige complete oplossing meer kan worden

- **ES** let niet op ondergrens/bovengrens voor waarde complete oplossing

  Bt let niet op ondergrens/bovengrens voor waarde complete oplossing (hoogstens op waarde huidige deeloplossing bij minimalisatieprobleem)

  B&B breidt deeloplossing niet verder uit als ondergrens/bovengrens voor waarde complete oplossing niet beter is dan beste waarde van tot nu toe gevonden complete oplossing

- **ES** houdt op elk moment slechts één deeloplossing bij (en impliciet de daaraan voorafgaande deeloplossingen in de toestandsboom)

  Bt houdt op elk moment slechts één deeloplossing bij (en impliciet de daaraan voorafgaande deeloplossingen in de toestandsboom)

  B&B houdt verzameling van deeloplossingen bij die (mogelijk) nog uitgebreid worden tot complete oplossing

- **ES** genereert bij uitbreiden van deeloplossing één kind tegelijk
  Bt genereert bij uitbreiden van deeloplossing één kind tegelijk
  B&B genereert bij uitbreiden van deeloplossing meteen alle kinderen

- **B&B** gebruikt ondergrens/bovengrens om deeloplossing te selecteren die als eerstvolgende wordt uitgebreid

- **ES** volgt (in de praktijk) depth-first wandeling in toestandsboom bij zoeken naar (optimale) oplossing
  Bt volgt depth-first wandeling in toestandsboom bij zoeken naar (optimale) oplossing
  B&B kan heen en weer springen in toestandsboom bij zoeken naar optimale oplossing

- **Lezen/leren bij dit college:** paragraaf 12.2
- **Werkcollege** Prim / Kruskal / branch & bound
  vanmiddag, 14.15–16.00
- **Opgaven:**
  zie `http://www.liacs.leidenuniv.nl/~vlietrvan1/algoritmiek`
- **Practicumbijeenkomst** programmeeropdracht 2:
  woensdag 4 mei 2021, 14.15-16.00, computerzalen Snellius
- **Daarna:** geen colleges meer, wel practicumbijeenkomsten
- **Laatste keer:** 10 of 17 mei: oud tentamen oefenen
- **Tentamen:** donderdag 9 juni 2022, 14.15–17.15
- **Vragenuur tentamen?** In overleg