Uge 46

Ex 1

1. Give an intuitive argument which shows that a graph $G = (V, E)$ is connected if and only if it has a spanning tree. Hint: how would you construct a spanning tree if the graph is connected?

Def of connected:

A graph $G=(V,E)$ is connected if it has a path from u to v for every choice of $u, v \in V$

Def of spannig tree:

A tree in a graph G is a connected subgraph $T=(V',E')$ of G with no cycle. It is a spanning tree if $V' = V$.

dvs. skal vise at alle grafer som er sammenhængende har et spanning tree.

- Starter med sammenhængende graf.
- Fjern kanter indtil, ingen cykler, dvs. bryd cykler.
    - Vi observerer at ved at fjerne en vilkårlig kant i en cykel vil grafen stadig være sammenhængende.
- Dvs. faktisk induktion fra sammenhængende graf til at vise man har spanning tree.
- Invariant, grafen er sammenhængende. Stop når ingen cykler -> stadig sammenhængende.

ex 2

2. Use the Greedy algorithm to find a minimum spanning tree in the graph from Figure 1.

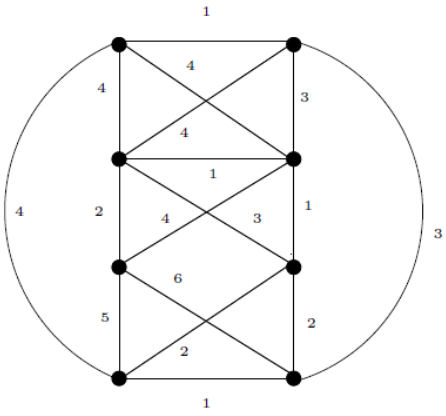
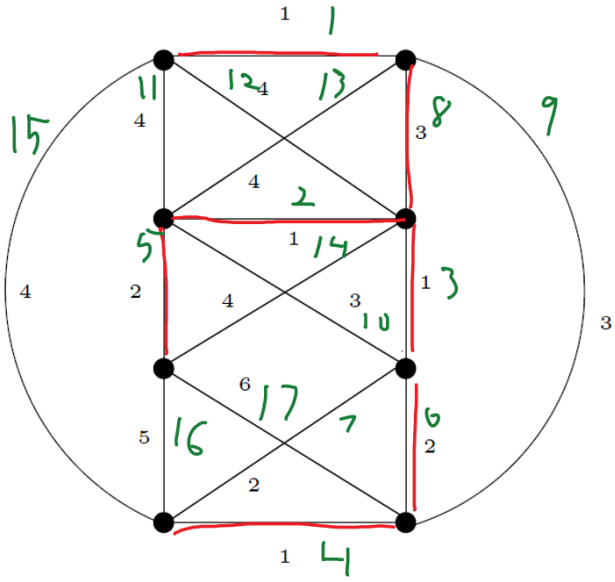


Figure 1: An edge weighted graph

(4) A greedy algorithm for the minimum spanning tree problem in $G=(V,E)$ (connected) with edge cost $c(e)>0$

G:

1. sort the edges according to their cost
   $c(e_1) \leq c(e_2) \leq \dots \leq c(e_{m-1}) \leq c(e_m)$ $m=|E|$
2. Set $X := \emptyset$
3. For $i := 1$ to m do
   If $H=(V, X \cup \{e_i\})$ has no cycle
   then $X := X \cup \{e_i\}$
4. Return $T=(V,X)$



Ex 3

3. Prove that if we add one new edge between two vertices u, v of a tree, then the resulting graph will have a cycle (comment: in fact one can show that it will have exactly one cycle).

- Vi ved at et træ er en sammenhængende graf uden cykler. Dvs. en graf hvor der er en sti fra ethvert valgt par af knuder.
- Samtidig ved vi at en cykel er defineret som en sti, hvor første og sidste knude er den samme.
- Dvs. når vi vælger en vilkårlig edge at tilføje, vil vi komme til at lukke en kreds, dvs. få en sti hvor første og sidste knude er den samme.

Ex 4

4. Let $G = (V, E)$ be a connected graph with a cost function c on its edges and suppose that the edge e has the lowest cost, that is, for every edge $e' \in E$ we have $c(e) \leq c(e')$.

   - Prove that e is contained in some minimum spanning tree of G. Hint: use the result from the exercise above to show how to construct a minimum spanning tree containing e: Take a minimum spanning tree T and assume it does not contain e. Then look at the cycle we get by adding e to T.
   - Can you conclude even more if we know that $c(e) < c(e')$ for all $e' \in E$ with $e' \neq e$?

- If we have a MST not containing e and add e, and remove any other edge (e') the total cost would be:
  - Cost before hand: $C(T)$
  - New cost when adding e and removing e' to get a ST again.:
    - $C(T') = C(T) + c(e) - c(e')$
    - due to $c(e') \geq c(e)$
    - $C(T') \leq C(T)$
    - meaning that e is in some spanning tree.
- if $c(e) < c(e')$ then it would be in all minimum spanning trees

Ex 5

5. Suppose $G = (V, E)$ is a graph with edge cost function c. Devise an algorithm to find a spanning tree of maximum cost. Suppose now that you have an algorithm $\mathcal{A}$ for minimum spanning tree and you are not allowed to build a new algorithm but you may define a new cost function c'. Show how to define c' so that we can solve the maximum cost spanning tree problem for G, c by using the algorithm $\mathcal{A}$ on G, c'. Hint: you may use the fact that every tree with n nodes has $n - 1$ edges (extra exercise: prove this), which in particular holds for every spanning tree on G if n is the number of vertices of G.

Alg to find maximum cost for graph G(V, E):

- sort edges according to cost (decreasing order)
- X ={}
- for each edge e:
    - if H=(V, X ∪ e) has no cycle:
        - X += e
- return T(V, X)

Define new cost function:

- Vi kan enten sætte $c' = -c$, hvis det er tilladt at have negativ kost.
- ellers: $c' = cmax - c$. dvs. vi forskyder så alle vægte er positive.

tree with n nodes → $n - 1$ edges (intuition).

- fix node.
- Skal være forbundet til alle andre noder ($n - 1$). Dvs. $n - 1$ edges.

Ex 6

6. For which of the following coin systems S does the greedy algorithm (always use as many coins of largest value as possible) work? If it does not work, you should give an example where it uses too many coins. If it works, you should try to prove this.

   (a) $S = \{10, 7, 2, 1\}$
   (b) $S = \{10, 7, 5, 2, 1\}$
   (c) $S = \{14, 10, 7, 5, 2, 1\}$
   (d) $S = \{20, 15, 10, 5, 2, 1\}$.

a. $n = 14 \rightarrow 10 + 2 + 2$, not optimal should have been $7 + 7$
b. *same*
c. $n = 20 \rightarrow 14 + 5 + 1$, not optimal should have been $10 + 10$
d. optimal:

proof:

following need to hold for every gready alg:

a. Greedy choice property:
A globally optimal solution can be obtaind by a series of locally optimal (greedy)choices

b. Optimal substructure property:
Every optimal solution contains optimal solutions to its subproblems.
After making a greedy choice a new (smaller problem of the same type arises)

We saw in the lecture that the Optimal substructure property holds for the coin changing problem. now we show that the GCP holds for this coinsystem:

Case 1: $n \geq 20$:

- Ser på $X = a_1 + \cdots + a_k$ som en optimal løsning, med flest 20 kr og grådig vil tage en 20 kr. mere.
- Dvs. denne optimale løsning har mindst en 20 kr mindre.
- Vi kan altså erstatte en 20 kr. fra den grådige med noget andet.
- hvilket betyder vi har $n' \geq 20$. Hvad kan vi erstatte med?
    - $2 \cdot 15$:
        - kunne tage $20 + 10$
        - MODSTRID, da vi skulle have den med flest 20 kr.
    - $15 + 10$:
        - kunne tage $20 + 5$
        - MODSTRID, da vi skulle have den med flest 20 kr.
    - $15 + 5$
        - kunne tage 20
        - MODSTRID, da vi skulle have den med flest 20 kr.
    - $15 + 2 + \cdots$
        - MODSTRID, vil ikke være en optimal løsning, da vi nu bruger flere mønter.

Case 2: $n < 20$ (ser nu på alle cases at den grådige gør det optimale:

- 19: $15 + 5$ fine
- 18: $15 + 2 + 1$ fine
- 17: $15 + 2$ fine
- 16: $15 + 1$ fine
- 15: 15 fine
- 14: $10 + 2 + 2$ fine
- 13: $10 + 2 + 1$ fine
- 12: $10 + 2$ fine
- 11: $10 + 1$ fine
- 10: 10 fine
- 9: $5 + 2 + 2$ fine
- 8: $5 + 2 + 1$ fine
- 7: $5 + 2$ fine
- 6: $5 + 1$ fine
- 5: 5 fine
- 4: $2 + 2$ fine
- 3: $2 + 1$ fine
- 2: 2 fine
- 1: 1 fine

Ex 7

7. Let $k \geq 3$ be an integer and consider the coin system $S = \{k+1, k, 1\}$. Show that on this system there is an amount for which the optimal solution uses 2 coins but the greedy algorithm will use k coins.

let $n = 2k$

example:

$$k = 4$$

$$n = 8 \rightarrow 5 + 1 + 1 + 1$$

optimally $k + k$

generally:

$$n = 2k$$

meaning we can only take one of the "$k + 1$"-coins:

$$2k - (k + 1) = 2k - k - 1 = k - 1$$

This means we cannot take any of the "k"-coins, and have to take the "1"-coins instead.

We have to take $k - 1$ of theses. Due to having taken one "$k + 1$"-coins, we have k coins. where we optimally just would have taken two of the "k" bills.

Ex 8

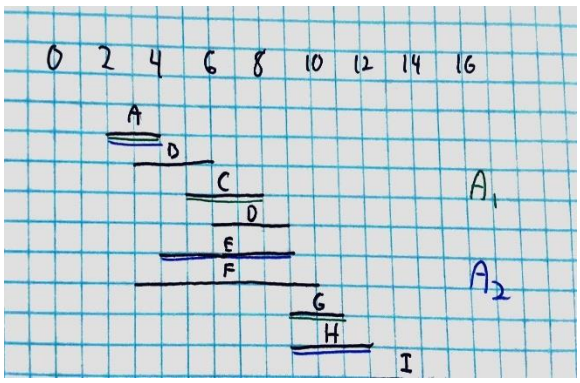
8. Suppose we have the following activities sorted according to their finishing time and with $C = [5; 8]$ meaning that activity C starts at time 5 and ends at time 8.

   - $A = [1; 4]$, $B = [3; 6]$, $C = [5; 8]$, $D = [6; 9]$, $E = [4; 9]$,
   - $F = [3, 10]$, $G = [9; 11]$, $H = [9; 12]$, $I = [12; 15]$.

   Your job is to find an optimal schedule for one room, that is, a schedule consisting of as many of the activities as possible and so that no two chosen activities overlap. Recall that this means that they cannot take place at the same time, but one may start just when the other finishes.

   - Find an optimal schedule using the greedy algorithm which always take the next activity as the one which has no overlap with the previously chosen ones and which ends as early as possible.
   - Try a different greedy strategy: always schedule that among the remaining activities which has no overlap with the already scheduled jobs and which has the largest starting time among such activities. Does it work? Can you prove that it always will work?

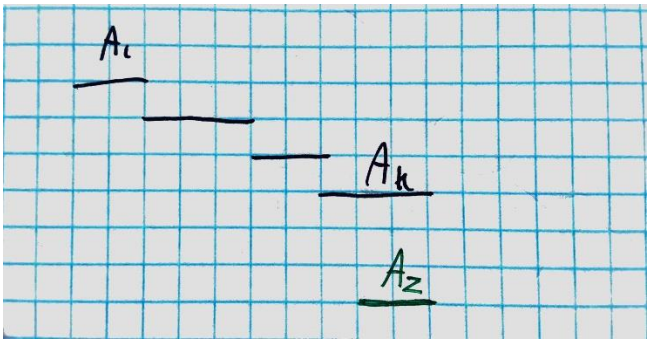
- A, C, G, I
- I, H, E, A



proof:

- we saw that the optimal substructor holds. Need to show that it has GCP.
- Need to show that there is a optimal solution which contains the greedy choice.

if we let $A_1 \dots A_k$ be an optimal solution (k activities):

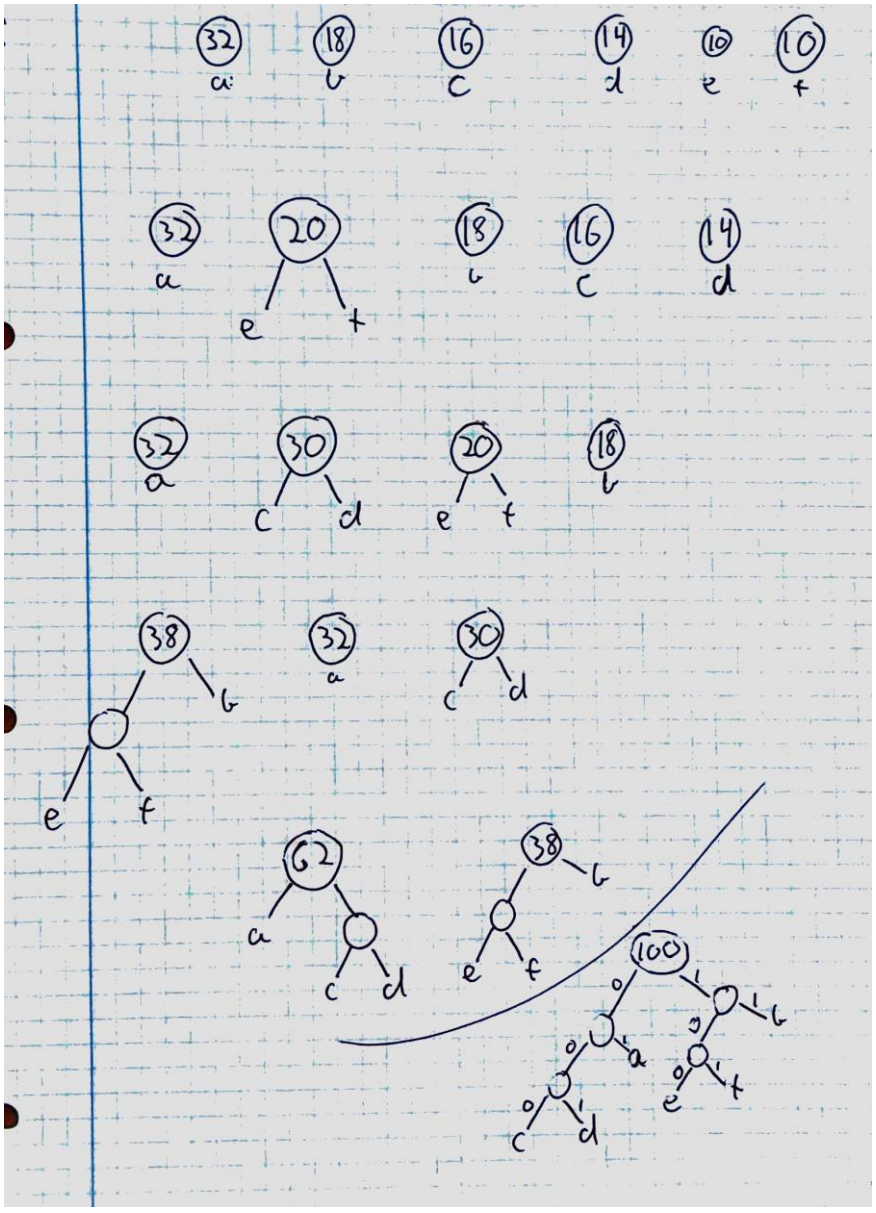


$s_z \geq s_k$, meaning that we can use A_z instead and get another optimal solution, now containing A_z

Observing that we can mirror the plan and get same number of activities we find that it automatically holds, due to proof shown in lecture.

Ex 9

9. Apply the Huffman coding algorithm for finding an optimal prefix code for a text with the following characters and frequences (in % of the total number of characters).
   $a:\ 32;\ b:\ 18;\ c:\ 16;\ d:\ 14;\ e:\ 10;\ f:\ 10$



A	01
B	11
C	000
D	001
E	100
F	101

Ex 10

10. Discuss what is needed if we want to implement Huffman's algorithm.

- Counting different chars
- Tree structure
    - Should be able to put these together.
    - Should be able to sort these according to total cost of leaves.
- Get codes -> traverse tree
- incode: translate from char to code
- decode: translate from code to char (traverse)

Ex 11

11. Let K_5 be the edge weighted complete graph in Figure 2.

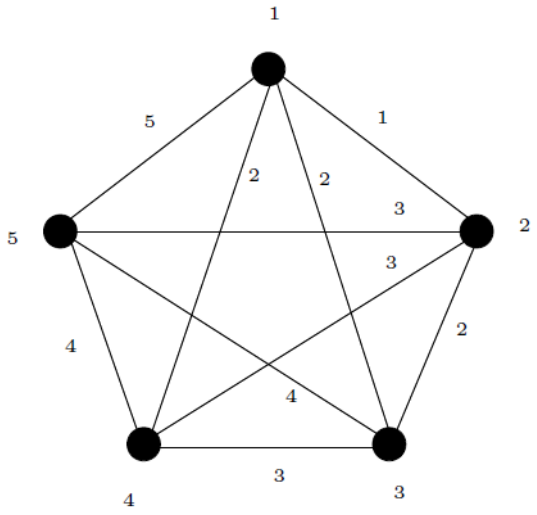


Figure 2: The graph K_5

- Construct a cycle on 5 vertices by starting in vertex number 1 and always including as the next vertex a vertex that is as close as possible (measured by the cost of the edge between the two vertices) to the last vertex we included (so the next vertex after 1 will be 2 in the example). What is the cost of the cycle?
- Is it an optimal solution? If not, produce one that is better.
- Can you change the cost of just one edge so that the cycle we find by the greedy strategy can be arbitrarily bad compared to the cheapest cycle?

- $1, 2, 3, 4, 5, 1 = 1 + 2 + 3 + 4 + 5 = 15$
- $1, 3, 5, 2, 4, 1 = 2 + 4 + 3 + 3 + 2 = 14$
  - 1,3,2,5,4,1 =13
- yes the last edge, linking 5 and 1, this have to be included at last, so setting this to an arbitrary large number will make the cycle arbitrarily bad.