CSE 201 2-25-21

Greedy Algorithms

Knapsack(v, w, W)

```
1.  n = length[v]
2.  for i = 1 to n
3.      x_i = 0
4.  weight = 0
5.  while weight < W
6.      i = (index of best remaining object)   { as measured by the selection function f(i)
7.      if weight + w_i ≤ W
8.          x_i = 1
9.          weight = weight + w_i
10.         mark i as included
11.     else
12.         x_i = (W - weight) / w_i
13.         weight = W
14. return x
```

Runtime:

worst case: $\Theta(n \log n)$

- either sort objects 1st

or • build a priority queue, then extract 'best':

$\Theta(n) + \Theta(n \log n) = \Theta(n \log n)$

Ex. discrete Knapsack: $n=5$, $W=10$

i	1	2	3	4	5
v	2	3	6.6	4	6
w	1	2	3	4	5
$\frac{v}{w}$	2	1.5	2.2	1	1.2

Greedy solution: $x = (1, 1, 1, 1, 0)$

value = 15.6, weight = 10

Exercise. Dyn. Prog. algorithm gives same solution.

Ex. $n = 5$, $W = 11$ (Discrete Knapsack)

i	1	2	3	4	5
v	1	6	18	22	28
w	1	2	5	6	7
$\frac{v}{w}$	1	3	3.6	3.67	4

Greedy solution: $x = (1, 1, 0, 0, 1)$

Value = 35, weight = 10

Exercise.

check that Dyn. Prog. yields

$y = (0, 0, 1, 1, 0)$

Value = 40, weight = 11

what about coin changing?

greedy algorithm:

- from amongst all denominations whose addition would not cause sum to exceed N, choose largest.
- stop when sum $\geq$ N.

denomination set: $d = (d_1, d_2, \ldots, d_N)$

assume: $d_1 < d_2 < d_3 < \ldots < d_N$

assume: $d_1 = 1$, so we can pay any N.

Exercise (hard)

show for $d = (1, 5, 10, 25, 50, 100)$ that greedy algorithm yields an optimal solution for any $N \geq 0$.

Exercise (easy)

show for $d = (1, 10, 25, 50, 100)$, the greedy algorithm doesn't yield OPT. soln. for some N.

Answer: $N = 30$.

Exercise (very hard)

characterize all denom. sets $d = (d_1, \ldots, d_n)$ for which greedy alg. yields an OPT. soln. (for all N.)

how to tell if a greedy alg. will work? Need.

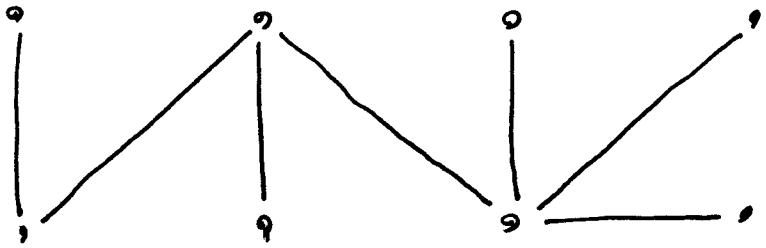
- Optimal substructure.
- Greedy choice Property:
  'local optimum is global optimum'
  i.e. a globally opt. soln. can be obtained by making a seq. of locally opt. choices (with respect to some selection function.)

Note: Selection fcn. might not coincide with objective fcn.

Minimum Weight Spanning Trees

Recall: a Tree is graph that is connected and acyclic.

Ex.



$n = 8$
$m = 7$

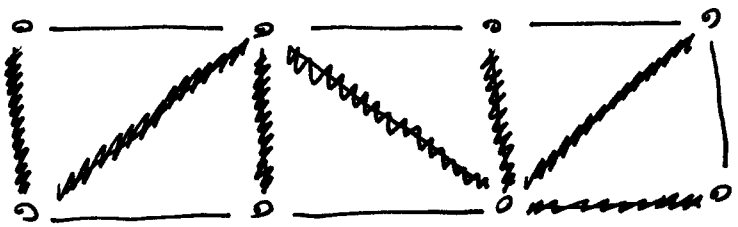
Thm The following are equivalent

- G is a tree
- G is connected and $m = n - 1$
- G is acyclic and $m = n - 1$

Here $n = |V(G)|$, $m = |E(G)|$

A subgraph of G is called a <u>spanning tree</u> iff (1) it is a tree, and (2) it includes all vertices.

<u>Ex</u>

G 

<u>Defn</u>

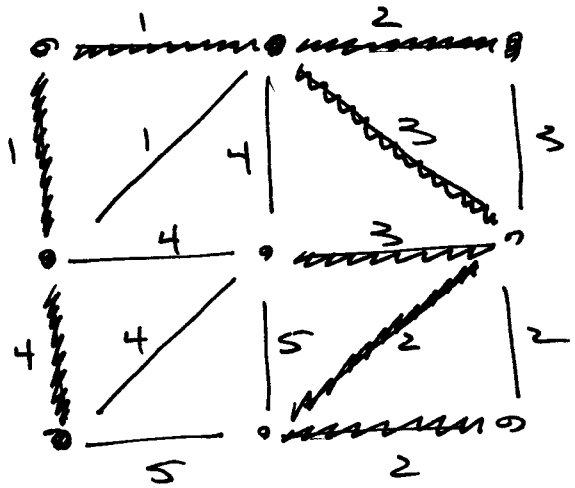
$G = (V, E)$ is a <u>weighted graph</u> if there is a non-negative weight fcn on edges

$$w : E \longrightarrow \mathbb{R}^+$$

Problem (MWST)

Determine a sp. tree of min. weight in G.

Ex.



$W(T) = 18$

↑

MWST

Two algorithms

- Kruskal's
- Prim's

} 23.2 in CLRS

Kruskal Algorithm

- choose an edge of min weight.
- amongst all edges whose addition would not create a cycle with previously chosen edges, choose one of min weight.
- stop when $n-1$ edges have been chosen.