

# Fusklapp Algorithms

Adam Sandberg Eriksson (910717-3099, saadam@student)

October 26, 2014

## Graphs

If graph problem and there is some constraint on the graph, it might be easier to first remove offending edges/nodes.

## Cut

Partition of vertices into two disjoint subsets.

## Cut-set

Any cut defines a cut-set: all edges that cross the cut.

## Min-cut

A cut is min if size of cut is  $\leq$  any other cut.

## Max-cut

A cut is max if size  $\geq$  any other cut.

## MST

### Cycle property

For any cycle  $C$  in the graph, if the weight of an edge  $e$  is larger than all other edges in  $C$ , then  $e$  does not belong to a MST.

### Cut property

For any cut  $C$ , if the weight of edge  $e$  in  $C$  is strictly smaller than all other edges of  $C$  then this edge belongs to all MSTs of the graph.

For any cut  $C$ , of graph  $G$ , the edge  $e$  included in MST of  $G$  is strictly smaller than all other edges in  $C$ .

## Minimum-cost edge

If edge  $e$  with minimum cost is unique it is included in all MST.

## Kruskal's algorithm

Consider edges in non-decreasing order. Add edge,  $e$ , to  $T$  if  $e$  does not introduce a cycle in  $T$ . Stop when all nodes are connected in  $T$ .

(Sort edges  $O(E \log E)$ , disjoint-set data structure to keep track of vertices are in which components.  $O(E \log V)$ ,  $O(V \log V)$ )

## Prim's algorithm

Add arb. node to  $T$ . Consider all edges out of  $T$ , grow  $T$  by the minimum-weight edge of these. Repeat until all nodes are in  $T$ .

(Adj matrix:  $O(V^2)$ . Bin. heap & adj list:  $O(E \log V)$ )

## BFS

Start with node, visit all connected nodes, repeat for each visited node. Can be done in  $O(E)$ . Can be used to find stuff in graph.

## DFS

Start with node, explore as far as possible along each branch before backtracking.  $O(E)$  for graphs traversed w/o repetition.

## Greedy algs

Think about: what ordering would change the total running time of algorithm. If some step takes constant time regardless of ordering do not consider this parameter in your greedy ordering.

## Exchange argument

Have your solution  $S$ , according to some greedy criterion, an arbitrary solution  $O$ . If  $S \neq O$  then there must be inversion (show what inversion is). Show that you can swap the inversion in  $O$  and not make  $O$  a worse solution.

Argument: if we swap all such inversions in  $O$  we will have  $S$  and thus  $S$  is no worse than an arbitrary solution and is therefore optimal.

## Stays ahead argument

Have your solution  $S$ , according to greedy

$$\begin{aligned} q' &= n - q = \sum_{i=1}^n (p_i + (1 - p_i)) - q \\ &= \sum_{i:l_i=0} p_i + \sum_{i:l_i=1} (1 - p_i) + \sum_{(i,j):l_i \neq l_j} S(i,j) \end{aligned}$$

Construct graph  $G = (V, E)$ .  $V = \{s, t\} \cup P$ ,  $P$  set of patients. For all  $v_i \in P$  we have edge  $(s, v_i)$  with capacity  $p_i$  and edge  $(v_i, t)$  with capacity  $1 - p_i$ . For each  $v_i, v_j \in P$  we have edge  $(v_i, v_j)$  with capacity  $S(i, j)$ .

## NP

To prove that a problem  $Q$  is NP-complete you have to:

1. Show that you can verify a solution to  $Q$  in polynomial time.
2. Reduce some NP-complete problem  $R$  to  $Q$ .
3. Show that a solution to  $R$  is a solution to  $Q$  and vice versa.

Page 498 has a list of NP-complete problems.

## Dynamic programming

Solve OPT backwards in recurrence. Iterate forwards in algorithm.

## Divide and conquer

### Useful

## Network flow

To show that solutions are equiv copy argument on page 412 solved exercise 2.

## Computing max flow from a min cut

Min cut: minimal weight cut, don't count incoming edges. All is well.

## Max/minimizing sums using NF

We have  $n$  patients, probability  $p_i$  of patient  $i$  having cancer, some measure of similarity between patients  $0 \leq S(i, j) \leq 1$ . We want to find labelling  $l_i \in \{0, 1\}$  of patients to maximize

$$q = \sum_{i:l_i=1} p_i + \sum_{i:l_i=0} (1 - p_i) - \sum_{(i,j):l_i \neq l_j} S(i,j)$$

We instead minimize

## Math

If  $a > 1, b > 1$  then  $a^{\log b} = b^{\log a}$ .

Geometric sum:  $\sum_{k=0}^{n-1} r^k = \frac{r^n - 1}{r - 1}$

Sum sum:  $\sum_{i=m}^n 1 = n + 1 - m$

Summelisummm:  $\sum_{i=m}^n i = \frac{(n+1-m)(n+m)}{2}$