Topic Review for Finals

# Gale-Shapley and Stable Matching

Full list of preferences, no ties. Gale-Shapley returns a perfect and stable matching. Men are matched to their best valid partners. Women are matched to their worst valid partners. The runtime is , being the number of men or the number of women, which are equal. Important things to know for proving stuff is that men go down their list of preferences, and women’s offers get better.

# Greedy Algorithms

## Unweighted Interval Scheduling

Input is a set of requests (time intervals). Goal is to return a set of non-overlapping intervals of maximum cardinality possible. The size of each interval doesn’t matter, we just want as many non-overlapping intervals as possible. Selecting earliest start time then deleting conflicts doesn’t work, selecting by increasing number of overlaps then deleting conflicts doesn’t work, selecting by increasing interval length and deleting conflicts doesn’t work. The algorithm is to select by **earliest finish time**! You can prove this by the exchange argument and induction.

## Minimum Spanning Tree

### Kruskal

Sort all edges in increasing order of cost. Pick the smallest edge. If it forms a cycle with the tree we have so far, throw it out. Else, include it. Repeat that until all nodes are included.

### Prim

Start from random node. Check outgoing edges of the tree we have so far, pick the one with the lowest cost. If it creates a cycle, throw it out. Else, keep it. Repeat that until all nodes are included. If all weights in a graph are unique, then there is a unique minimum spanning tree, so all MST algorithms returns the same thing. Just a useful thing to know, the time complexity of DFS is .

### Borůvka

Initialize all vertices to be a component. While there is more than one component (i.e. no spanning tree yet), find the cheapest edge for each component and add it to our current trees (this is done in parallel for all components). Requires distinct edge weights. Because imagine a triangle with the same weight on each edge. Each one can choose the edge to the left and we would end up with a cycle.

## Huffman Coding

Not done in homework or practice sets. Do not need to know proof. Just need to know reduction to other problems.

We want binary representations for each letter in a lexicon, want shorter strings for more frequently used letters and longer strings for the less frequently used ones. Also! (This is important.) We don’t want a letter’s coding to be the prefix to another letter’s coding, i.e. if a’s code is 10, then other letter’s code can’t start with 01, like c can’t be 101. Another objective is to have the average length of all codes as short as possible.

# Dynamic Programming

## Weighted Interval Scheduling

meow

## Segmented Least Squares

Not done in homework or practice sets. Do not need to know proof. Just need to know reduction to other problems.

## Sequence Alignment

Not done in homework or practice sets. Do not need to know proof. Just need to know reduction to other problems.

## Bellman-Ford

meow

# Divide and Conquer

## Integer Multiplication

meow

## Convolution

Only to recognize a convolution problem, not details of how to run the algorithm.

# Randomized Algorithms

## Linear-Time Median Finding

meow

## Hashing

meow

## Prime Testing

# Network Flow

## Ford-Fulkerson

meow

## Min Cut

meow

## Applications

meow

# NP-Completeness

## SAT problem

meow

## Independent Set

meow

## Vertex Cover

meow

## Set Cover

meow

## Hamiltonian Path

meow

## Traveling Salesman Problem

meow

# Computability

## Halting Problem

meow

## Co-Halting Problem

meow

## Accept Problem

meow

# Approximation Algorithms

## Greedy Methods

### Knapsack

meow

## Linear Programming

### Vertex Cover

meow