This document lists the things that you should be able to do for the first prelim.

1 Problem sets

Questions about problem sets 1 and 2 are fair game. Be sure you can explain and can avoid any mistakes you made on them.

2 Stable matching

Definitions:

• recall the definitions of matching, perfect matching, stable matching, \( best(m) \), \( worst(m) \)

Sample question: Define stable matching.

Sample question: Give an example stable matching problem instance having some \( m \) for whom \( best(m) \) is not \( m \)’s first choice.

Sample question: Is the following matching stable? [picture]

Sample question: Find \( best(m_1) \) in the following matching problem. [picture]

Gale-Shapley Algorithm

• describe and execute the Gale-Shapley algorithm

Sample question: List the engagements that take place while executing Gale-Shapley on the following instance of the stable matching problem. [picture]

• characterize the result of the algorithm

Sample question: Can the following matching be produced by the Gale-Shapley algorithm? Why or why not?

3 Greedy algorithms

Problems:

• describe the greedy algorithms we gave in class
  
  – Interval scheduling / first end time
  
  – Scheduling to minimize lateness / first deadline
- MST / Prim’s Algorithm, Kruskal’s Algorithm

**Sample question:** Recall that the definition of the scheduling with minimum lateness problem is [...]. Describe a greedy algorithm that produces an optimal solution to this problem.

**Sample question:** Give pseudocode for Prim’s algorithm

**Sample question:** List the edges added to the minimum spanning tree in the order they are added by Kruskal’s algorithm

Proof techniques

- construct proofs using a “greedy stays ahead” argument

**Sample question:** Consider the following problem: [...]. Suppose Alice has given you the following algorithm: [...]. Suppose that Alice has also proved that at every step, her algorithm is ahead of an optimal solution, in the sense that \( f(e_i) \leq f(o) \). Show that Alice’s algorithm produces an optimal solution.

**Sample question:** Consider the following problem: [...]. Suppose Bob proposes the following greedy algorithm to solve it. By what measure does his algorithm “stay ahead”? Give an inductive proof that ther is true.

Greedy algorithm design

- Propose greedy algorithms for problems

**Sample question:** Consider the following problem: [...]. Give a greedy algorithm that produces an optimal solution to this problem.

- Construct counterexamples for greedy algorithms

**Sample question:** Consider the following problem: [...]. Chuck claims that the following algorithm produces an optimal result. Construct an example where Chuck’s algorithm produces an incorrect result.

4 Divide and conquer

Analysis techniques

- Informal analysis by drawing recursion trees

**Sample question:** Consider the following divide-and-conquer style algorithm: [...]. Write a summation describing the running time of the algorithm.

- Derive a recurrence relation from an algorithm
Sample question: Let $T(n)$ be the running time for the following algorithm: [...]. Write a recurrence relation that describes $T(n)$.

- Construct an inductive proof of a recurrence solution

Sample question: Suppose an algorithm’s running time satisfies $T(n) \leq 2T(n/2) + n^2$. Assume that $C > T(1)$ and $C > 2$. Prove by induction that $T(n) \leq Cn^2$.

5 Dynamic programming

Problems:

- Describe and execute the dynamic programming algorithms we described in class
  - Weighted interval scheduling
  - Sequence alignment
  - Shortest paths with negative edges / Bellman-Ford

Sample question: Recall the definition of the weighted interval scheduling problem: [...]. Describe a dynamic programming solution to this problem.

Sample question: Fill in the following memo table for an execution of the Bellman-ford algorithm on the following graph: [...].

Proof techniques:

- Given a problem description, describe the form of an optimal solution in various cases.

Sample question: Consider the following problem. Suppose you are given a weighted set of objects $o_1, o_2, \ldots, o_n$ with weights $w_1, w_2, \ldots, w_n$, and a weight budget $W$. Describe the possible solutions that contain $o_n$. Describe the optimal solutions that do not contain $o_n$.

- Given a recursive program to compute a solution, construct suitable memo tables for that solution and give the running time the resulting dynamic program.

Sample question: Give an upper bound on the running time for a memoized version of the following algorithm: [...]