Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Huffman Proofs

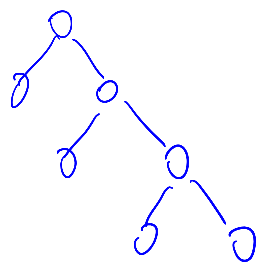
(a) Prove that in the Huffman coding scheme, if some symbol occurs with frequency more than $\frac{2}{5}$, then there is guaranteed to be a codeword of length 1. Also prove that if all symbols occur with frequency less than $\frac{1}{3}$, then there is guaranteed to be no codeword of length 1.

(b) Suppose that our alphabet consists of n symbols. What is the longest possible encoding of a single symbol under the Huffman code? What set of frequencies yields such an encoding?

(a)

$n-1$

(b)

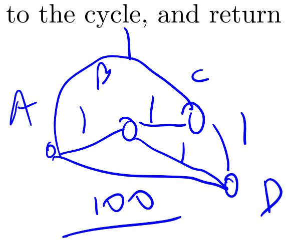


2 Finding Counterexamples

In this problem, we give example greedy algorithms for various problems, and your goal is to find an example where they are not optimal.

(a) In the travelling salesman problem, we have a weighted undirected graph $G(V, E)$ with all possible edges. Our goal is to find the cycle that visits all the vertices exactly once with minimum length.

One greedy algorithm is: Build the cycle starting from an arbitrary start point s, and initialize the set of visited vertices to just s. At each step, if we are currently at vertex u and our cycle has not visited all the vertices yet, add the shortest edge from u to an unvisited vertex v to the cycle, and then move to v and mark v as visited. Otherwise, add an edge from the current vertex to s to the cycle, and return the now complete cycle.

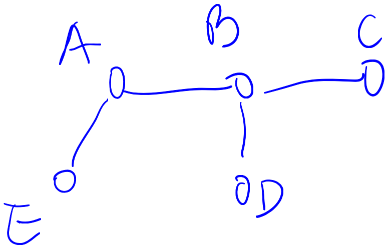


$A \to B \to C \to D$ is not optimal

$A \to B \to D \to C \to A$

(b) In the maximum matching problem, we have an undirected graph $G(V, E)$ and our goal is to find the largest matching E' in E, i.e. the largest subset E' of E such that no two edges in E' share an endpoint.

One greedy algorithm is: While there is an edge $e = (u, v)$ in E such that neither u or v is already an endpoint of an edge in E', add any such edge to E'. (Can you prove that this algorithm still finds a solution whose size is at least half the size of the best solution?)



first add AB will not be optimal

3 Service scheduling

A server has n customers waiting to be served. Customer i requires t_i minutes to be served. If, for example, the customers were served in the order $t_1, t_2, t_3, \dots, t_n$, then the i-th customer would wait for $t_1 + t_2 + \cdots + t_i$ minutes.

We want to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i) .$$

Given the list of the t_i's, give an efficient algorithm for computing the optimal order in which to serve the customers.

order by time, from smaller to bigger.
this is the service order.

4 Finding MSTs by Deleting Edges

Consider the following algorithm to find the minimum spanning tree of an undirected, weighted graph $G(V, E)$. For simplicity, you may assume that no two edges in G have the same weight.

procedure FINDMST($G(V, E)$)
  $E' \leftarrow E$
  for Each edge e in E in decreasing weight order **do**
    if $G(V, E' - e)$ is connected **then**
      $E' \leftarrow E' - e$
  return E'

Show that this algorithm outputs a minimum spanning tree of G.