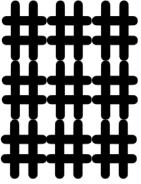


CS5112 Final Exam

December 4, 2018

This test is closed book and closed notes. You have the entire class period, 75 minutes. There is an additional blank page at the end that you can use if you run out of space for your answers.

You are welcome to use the backs of pages for scratch. However, we will not grade anything you write on the back of any page, including the blank one at the end.

We also will not look at anything written in the top left corner over the hash mark. Be sure that your answer is not ambiguous – if you write multiple solutions, there are no promises which one we will grade.

Point values are given at the top of each page, and in [square brackets] at the start of each question. Take care to answer all the questions. We have used the symbol $\Longrightarrow$ to indicate that an answer is required, and questions that should be true or false are marked T/F.

Note that the very last problem is only worth 6 points but is much harder than any of the other problems. Be sure to completely finish the other problems before working on it. It is marked with a black diamond: ♦

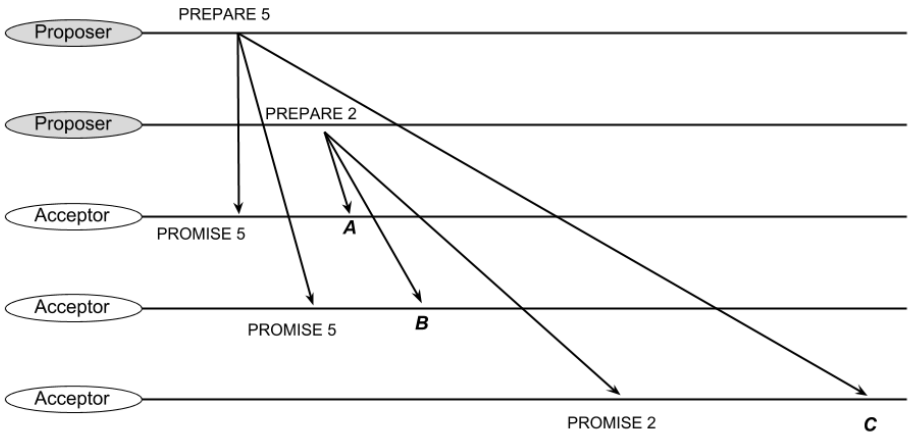
WRITE YOUR NAME BELOW. WRITE YOUR NETID AT TOP RIGHT OF EACH PAGE (not your name, e.g. “rz13” not “Ramin Zabih”)

_____

NAME

Problem 1: Paxos [9 points]

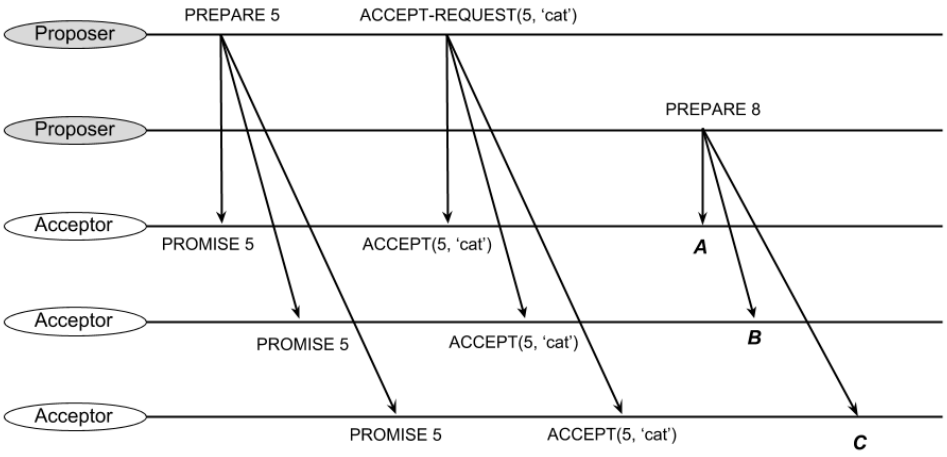
The following questions reference diagrams of PAXOS runs in progress. For each, note that the horizontal lines are timelines (the future is further to the right). The diagonal arrows from timeline to timeline represent requests, with the label at the beginning of the arrow specifying what the Proposer sends to the Acceptors, and the label at the end of the arrow specifying the Acceptor's response.



Referring to the diagram above, answer the following 2 questions.

1. [3] Which of the following responses should be at label **A**? ⟹
   a. PROMISE 5
   b. PROMISE 2
   c. PREPARE 5
   d. None - the request is ignored

2. [3] Which of the following responses should be at label **C**? ⟹
   a. PROMISE 5
   b. PROMISE 2
   c. PREPARE 5
   d. None - the request is ignored



Referring to the diagram above (not the one on the previous page), answer the following question.

3. [3] Which of the following responses should be at label ***B***? $\Longrightarrow$
   a. PROMISE 8
   b. ACCEPT-REQUEST(8, 'cat')
   c. PROMISE 8, ACCEPTED 'cat'
   d. None - the request is ignored.

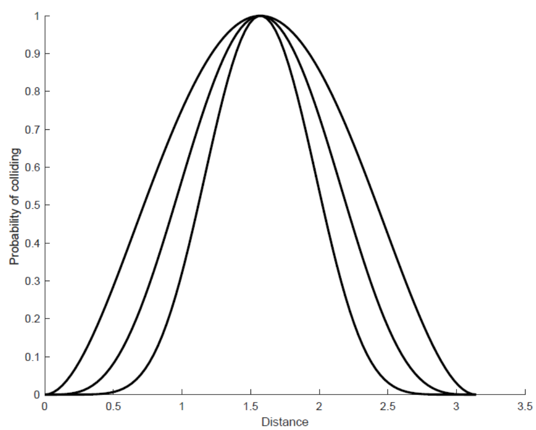
Problem 2: LSH [15 points]

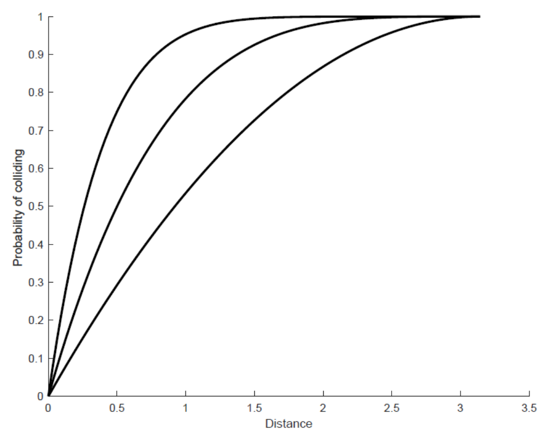
This problem is about SimHash, locality-sensitive hashing on vectors using random projections. In this context, the distance between two input vectors x and y is the angle between them. So for example two nearly parallel vectors are very similar, even if their magnitudes are very different.

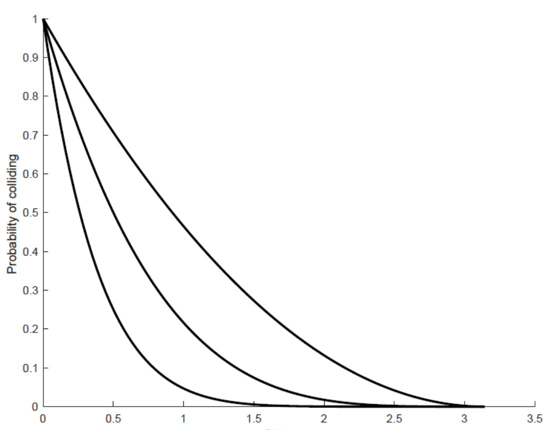
As a reminder, producing a k-bit, random SimHash hash function h is created by choosing k random vectors $r_1, \ldots r_k$. Then for any input vector x, the i^{th} bit of $h(x)$ is given by

$$\begin{cases} 0 & \text{if} \quad x \cdot r_i >= 0 \\ 1 & \text{otherwise} \end{cases}$$

Plotted in one of the three below plots is the probability that two vectors x and y end up with the same hash value, as a function of the angle between them. This function has been plotted for $k = 2$, $k = 4$, and $k = 8$. The other two plots are incorrect. Circle the correct plot, and on the plot label each of the 3 curves with its corresponding k value. $\Longrightarrow$

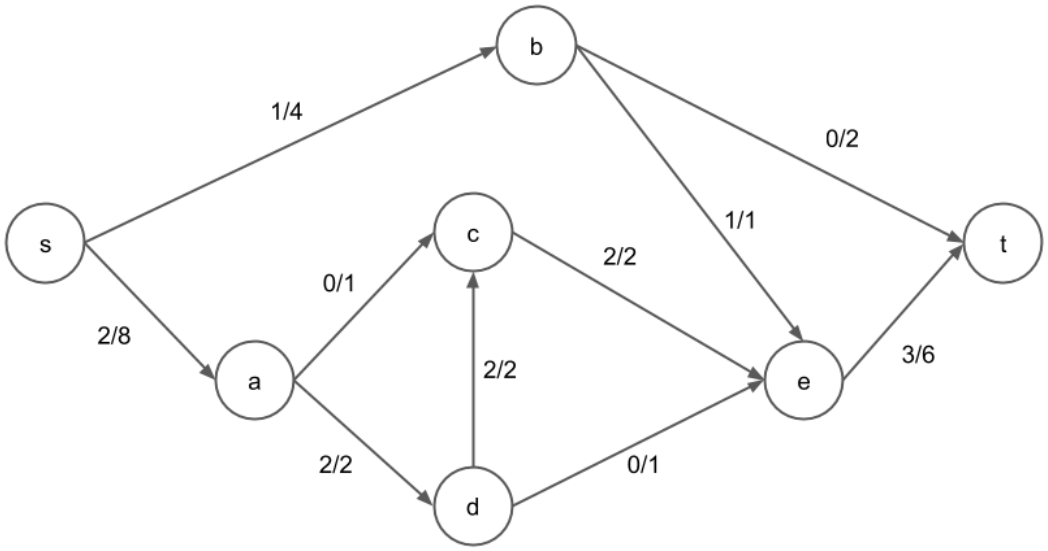




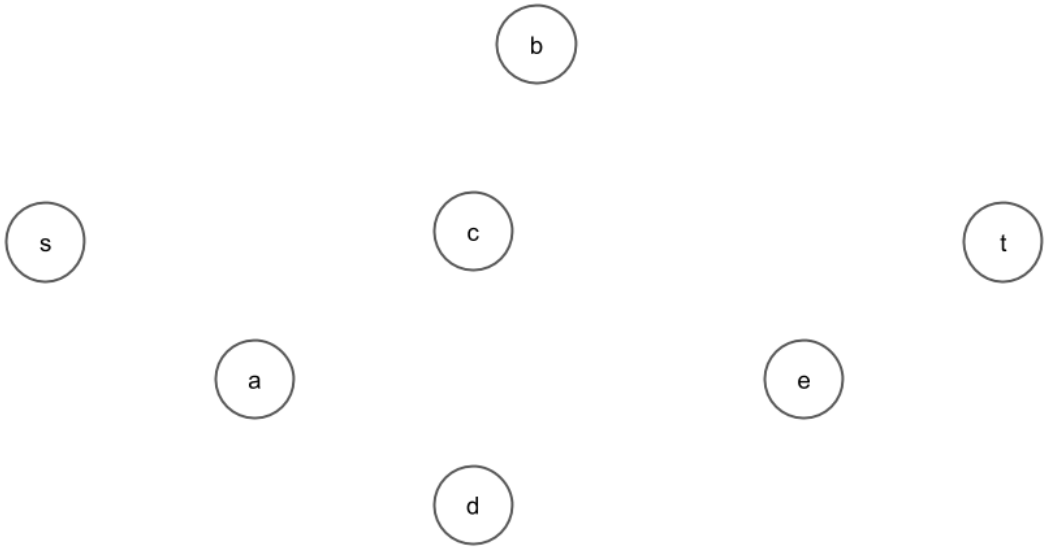


Problem 3: Max Flow / Min Cut [20 points]

Consider the following flow diagram (with *s* as the source and *t* as the sink), with some flow already running through it (denoted as *f/c* where *f* is the flow and *c* is the capacity):

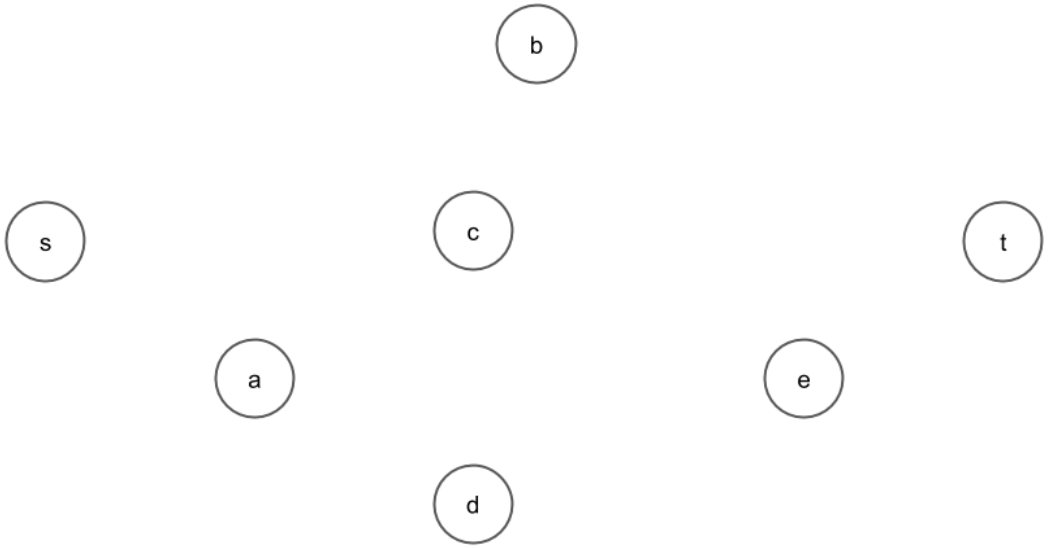


a) [5] Draw the corresponding residual graph below (the nodes in the graph are provided for you) ⟹



b) [5] What is the Max Flow in this graph? ⟹

c) [5] Draw the residual graph when the flow through the graph is maximized (the nodes in the graph are provided for you) ⟹



d) [5] Recall that an *s-t cut* of a graph is a partition of the graphs nodes into two sets *S* and *T* such that node *s* is a member of *S* and that node *t* is a member of *T*. Recall further that the *capacity* of such a cut is the sum of the capacities of the edges in the graph that go from a node in *S* to a node in *T*. The *"min cut"* is the *s-t cut* with minimum capacity. Give the *min cut* of the graph above by specifying which nodes are in *S* and *T*. ⟹

S = { *s*, }

T = { *t*, }

Problem 4. Dynamic programming [25 points]

In the change making problem, a shopkeeper owes a customer some change, and has a limited supply of coins in the cash register. We are given:

- A *target*: an integer $k > 0$, and
- A list of *available coins*: $C = [c_1, c_2, \ldots, c_n]$. These are the coins in the register. There are n of them, and $c_i > 0$ is an integer.

We should output True or False according to whether the shopkeeper can produce exactly *k* cents using the coins in the register.

For example, suppose we have two 10-cent coins, four 5-cent coins, and a 20- cent coin, and we owe 45 cents. Then k = 45, n = 7 and C = [10, 10, 5, 5, 5, 5, 20]. The answer is True: one way to make this change is with a single ten-cent coin, the twenty-cent coin, and three of the five-cent coins.

We will solve this problem using dynamic programming. We are only interested in whether it is possible to produce *k* cents – we do not need to return the actual set of coins used. We will use a two-dimensional table of size $k \times (n + 1)$. The cell $T(i, j)$, for $1 \leq i \leq k$, $0 \leq j \leq n$ holds the true/false value of whether it is possible to make i cents using only the first j coins, that is, $[c_1, \ldots, c_j]$. Note that $j = 0$ means using no coins.

Fill in the recurrence below:

$$T(i,j) = \begin{cases} \text{False} & \text{if } j = 0 \\ \text{True} & \text{if } j > 0 \text{ and } c_j = i \\ \textbf{Part A} & \text{if } j > 0 \text{ and } c_j > i \\ \textbf{Part B} & \text{if } j > 0 \text{ and } c_j < i \end{cases}$$

Each of Part A and Part B should make reference to at most two other table cells.

Part A: [10] $\Longrightarrow$

Part B: [15] $\Longrightarrow$

Problem 5: Union-Find [25 points]

For parts a-c, consider the following array representation of a forest:

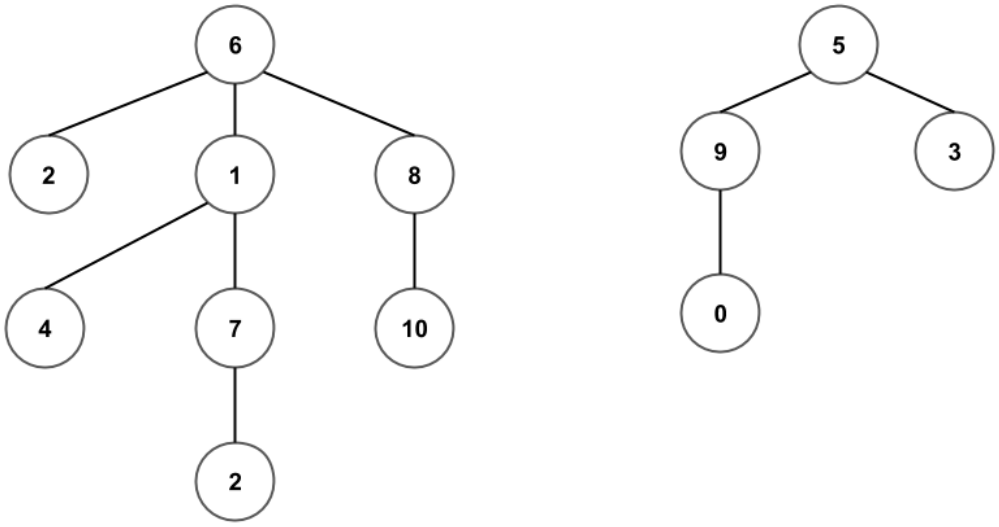
3	2	0	3	5	5	1	9	8	5

a) [7] Draw the corresponding forest below: $\Longrightarrow$

b) [3] How many connected components are there in the forest above? $\Longrightarrow$

c) [3] True or false: nodes *5* and *6* are members of the same connected component. $\Longrightarrow$

d) [12] Consider the following forest:



Assume that Union-Find utilizes both *path-compression* ((flattening the tree to make future finds faster) and *union-by-size* (minimizing how many nodes are impacted by the union) optimizations. Suppose we call *union(0, 7)* on this forest. Give its resulting array representation. $\Longrightarrow$

♦ **Problem 6.** Dynamic programming [6 points]

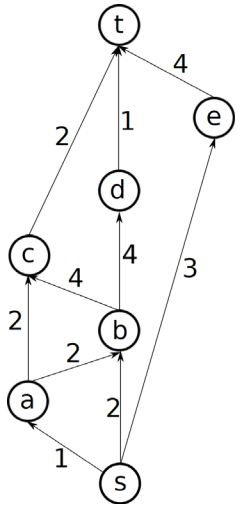
The ski lifts at mount 5112 have sadly broken down, but Alys and Bert are determined to ski nonetheless. Unfortunately, they need to carry their skis up the mountain.

The paths up the mountain are given as a directed, acyclic graph $G = (V, E)$. Each vertex v is a place where they can stop; paths run between these resting points. Each resting point has a height $h(v)$. Paths only go uphill, that is, if (u, v) is an edge then $h(u) < h(v)$.

Each edge (u, v) has a length $l(u, v)$. The vertex s, at the bottom of the mountain, is the vertex for the base where they set out from; t, at the top, is the vertex for the summit. They will walk up, and along each edge, either Alys or Bert will carry the skis.

They would like to know whether there is a path from s to t, along with an assignment of edges on that path, so that Alys and Bert each carry the skis for the same total length. We will call such a path *fair*.

For example, here is a graph labelled with edge lengths:



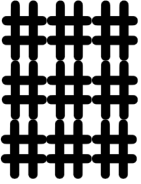
This graph has a fair path: $s \to a \to b \to d \to t$ has total length 8; if for example Alys carries the skis for $b \to d$ then each of them carries for a length distance of 4.

We will solve this problem using dynamic programming. For this problem, we are only interested in the true/false value of whether there is a fair path; don't worry about actually computing the path. If W is the sum of all edge lengths in G and n the number of vertices, we will use a table of size $(2W + 1) \times n$, which we will call $T(x, v)$. Each element of the table is a Boolean. It is up to you to come up with the optimal substructure leading to such a table.

Part A [2] Write down a short English description of $T(x, v)$. (Hint: it may be easier to think of $-W \leq x \leq W$ rather than $0 \leq x \leq 2W + 1$) $\Longrightarrow$

Part B [3] Write down a recursive definition for T. $\Longrightarrow$

Part C [1] In what order should the cells be filled? $\Longrightarrow$



“Computers are useless, they can only give you answers.” - Picasso