CSE 417 : Winter 2021 Final Exam

Directions

- This is a take home exam. It is due on Wednesday March 17 at 11:59 PM (Seattle time, i.e. GMT-7).
- Remember to follow the collaboration policy:
  - You are permitted to work in groups of up to 3 other students in the class; you may discuss the exam only with those students and the course staff.
  - As always, you must write up your work independently.
- This exam is open-book/open-note. You may use any resource we have provided, as well as external resources.
- But you may not search for the particular problems in the exam.
- If you accidentally find solutions to the problem we asked online, you must tell us (via private Ed post). There will be no penalty for disclosing you accidentally discovered a solution.
- Submit your responses to gradescope. You may handwrite or typeset solutions (as long as they are legible). We do not recommend trying to write on this document; we have not left enough room.
- If you have a question, please check the pinned post on Ed where we've posted common clarifications. If you still have a question, please ask a **private** question on ed.

Problem Selection

- The exam is broken into 4 sections. You will do all subparts of **one** *numbered* problem in each of the sections. For simplicity, we've put every numbered question on its own page. So, in each of the 4 sections, you choose one page to do.
- When you submit to gradescope, there will be a question for both options in a given section. You will select a page for only one of those questions.
- If you submit two problems for a single section, we will flip a coin and grade only the one chosen by coin flip.

Question	Max points
Dynamic	20
Graphs	20
The [NP]-Hard[ness] Part	20
Mystery	20
Survey	1
Total	**81**

1. Dynamic

In this section you will design a dynamic programming algorithm. Do one of the two problems in this section.

1.1. Subarrays

Recall that a **contiguous subarray** is all of the elements in an array between indices i and j, inclusive (and if $j < i$ we define it to be the empty array).

Call a subarray is **nearly contiguous** if it is contiguous (i.e. contains all elements between indices i and j for some i, j) or if it contains all but one of the elements between i and j.

For example, in the array $[0, 1, 2, 3, 4]$,

- $[0, 2, 3]$ is nearly contiguous (from 0 to 3, skipping 1), sum is 5
- $[0, 1, 2, 3]$ is nearly contiguous (because it is contiguous from 0 to 3), sum is 6.
- $[2, 4]$ is nearly contiguous (from 2 to 4, skipping 3), sum is 6.
- $[3]$ is nearly contiguous (because it is contiguous from 3 to 3), sum is 3.
- $[]$ is nearly contiguous (because it is contiguous from 1 to 0), sum is 0.
- $[0, 2, 4]$ is **not** nearly contiguous (because you'd have to remove two elements).

The sum of a nearly contiguous subarray is the sum of the included elements.

Given `int[] A`, your task is to return the maximum sum of a nearly contiguous subarray.

For example, on input $[10, 9, -3, 4, -100, -20, 15, -5, 9]$ your algorithm should return 24 (corresponding to $i = 6, j = 8$ and skipping the -5).

(a) Define one or more recurrences to solve this problem.

(b) Give English descriptions (1-2 sentences each should suffice) for what your recurrences calculate. Be sure to mention what any parameter(s) mean.

(c) How do you caclulate your final overall answer (e.g. what parameters input to which of your recurrences do you check).

(d) What memoization structure(s) would you use?

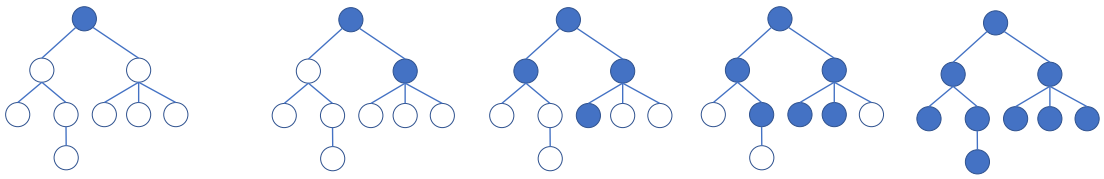
(e) What would the running time of your algorithm be (you do **not** have to write the code). Justify in 1-3 sentences.

1.2. An Important Annoucnement

The CEO of your company has **great** news, too great to share in an email. Instead the CEO wants direct supervisors to call each of their direct reports and personally tell them the news. But the news is so great, the CEO wants it to go out as quickly as possible. They want your help figuring out who needs to call who.

Formally, you have a rooted tree T. Announcing the news occurs in rounds. At the start of round 1 only the root knows the news. In every round, every node that knows the news chooses one of its children and shares the news with that child (if all of its children know the news, it no longer does anything). Your goal is to determine the minimum number of rounds it will take to spread the news to every node.

For example, in the tree below we have shown a minimum number of rounds. The correct answer for this tree is 4.



The leftmost tree is the state before the first round (the root knows the news [colored in], but no other vertex knows [not colored in]). The next four graphs are the states after each of the 4 rounds.

(a) Define one or more recurrences to solve this problem.

(b) Give English descriptions (1-2 sentences each should suffice) for what your recurrences calculate. Be sure to mention what any parameter(s) mean.

(c) How do you caclulate your final overall answer (e.g. what parameters input to which of your recurrences do you check).

(d) What memoization structure(s) would you use?

(e) What would the running time of your algorithm be (you do **not** have to write the code). Justify in 1-3 sentences.

The Dynamic section ends here.

2. Graphs

In this section, you will design an algorithm that uses a graph as a fundamental piece of the algorithm design process. Do one of the two problems in this section.

2.1. World of Pathcraft

You are playing a strategy (video) game. You have k characters at k different cities on the game map. The game map is defined as a set of cities connected via one-way roads.

You decide to use all k of your characters to attack the enemy's capital city (which involves moving them all through the map to that city). But there's a catch – the AI you're playing against will close down a road once they realize one of your characters has used it.

Your goal, therefore, is to send all characters from their current locations to the enemy capital without any two characters using the same road.[1]

Input:

- A list of the k cities where your characters are located.
- The identity of the enemy capital.
- The game map

Output:
`True` if you can send all k characters to the enemy capital without any of them reusing a road.
`False` if you cannot.

(a) What are the vertices of the graph you'd like to use for this problem?

(b) What are the edges? (you might want to include: are they directed or undirected? if you have edge weights what are they?)

(c) How will you find whether it's possible to send all k characters to the capital? You only need to say "Yes it's possible" or "No, it's not possible" (you do not need to record the paths themselves).

(d) Give a 2-3 sentence explanation of why the previous part produces the right output (this does not need to be a full proof, but be sure to mention both "Yes" and "No" cases).

(e) What is the running time of your algorithm? (Justify with 1-2 sentences)

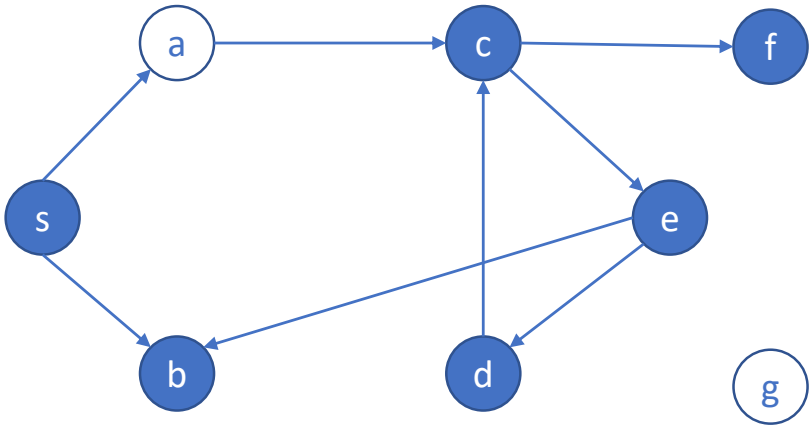
[1] Meeting at an intermediate city is fine, as long as the two characters enter using different roads and leave using different roads.

2.2. Put One Foot In Front of The Other

Recall that a **walk** in a graph is a list of vertices $v_0, v_1, \ldots, v_k$ such that there is an edge from v_i to v_{i+1} for every i. A walk is allowed to repeat both vertices and edges. Call the **length** of a path the number of edges included in it (counting repeats as many times as they appear).

For a starting vertex s, call u "**even-reachable**" if there is a walk from s to u of even length.

For example, in the graph below the colored in vertices are even-reachable.



s is even-reachable (never leaving is a walk with 0 edges).
b is even-reachable (s, a, c, e, b has 4 edges. Note we don't need the shortest route to be even, just a route)
c is even-reachable (s, a, c has 2 edges)
d is even-reachable (s, a, c, e, d has 4 edges)
e is even-reachable (s, a, c, e, b, c, e has 6 edges. Note that we are allowed to go around the cycle repeatedly to make the length even)
f is even-reachable (s, a, c, e, d, c, f has 6 edges)

a is not even-reachable (from s the only walk has one edge)
g is not even-reachable (it cannot be reached from g at all)

(a) Write pseudocode (or English) for an algorithm, given a directed graph G and a starting vertex s to check whether each vertex is even-reachable from s. You may assume that every vertex has any fields to help you write an algorithm. Store the answer for `v` in the `boolen` field `v.evenReach`.

(b) What is the running time of your algorithm? An $\mathcal{O}(m+n)$ algorithm will receive full credit (where G has m edges and n vertices). Slower algorithms will receive partial credit.

(c) Briefly (2-3 sentences) explain how your algorithm works.

The Graphs section ends here.

3. The [NP-]Hard[ness] Part

In this section, you will prove a problem NP-hard. Do one of the two problems in this section.

3.1. Double-3-SAT

Call the following problem "DOUBLE-3-SAT"

Input: A list of boolean variables and a list of constraints in standard 3-SAT form.
Output: `True` if there are **at least** two different settings of the variables that will satisfy all constraints.
`False` otherwise.

So DOUBLE-3-SAT is nearly identical to standard 3-SAT, except that you need two different settings of the variables, not just one.

(a) Give the pseudocode (or English description) of a reduction to show that DOUBLE-3-SAT is NP-hard.

(b) Argue that your reduction is correct (remember to handle both directions). Our descriptions were 2 sentences each.

3.2. Almost Coloring

For an undirected (and unweighted) graph G, we say that G is Almost-3-colorable if you can assign one of the labels $1, 2, 3$ to each vertex of G so that at most one edge has both of its endpoints with the same label.

The ALMOST-3-COLORABLE problem is:
Input: An undirected graph G
Output: `True` if G is almost-3-colorable (i.e., there is a way to label the vertices $1, 2, 3$ so that at most one edge has both of its endpoints with the same label)
`False` otherwise

(a) Give the pseudocode (or English description) of a reduction to show that ALMOST-3-COLORABLE is NP-hard.

(b) Argue that your reduction is correct (remember to handle both directions). Our descriptions were 2 sentences each.

The NP-hardness section ends here.

4. Mystery

Choose only one question from this section. This section has no theme – You might find any of the techniques from class useful. For some problems there may even be more than one of our techniques that works.

4.1. Stonks

You've been assigned to write an article about the recent increase in the value of Gamestop stock. You wish to find the price changes that will be most impressive to your readers. Specifically, you have a list of the prices of the stock at the end of n consecutive days (stored in an array A).

You want to find a **contiguous** subarray where the stock is only going up and (subject to the only increasing condition) the increase is as large as possible. More formally, we want to find a subarray defined by indices i, j such that

- The price is increasing every day, i.e., $A[i] < A[i+1] < A[i+2] < \cdots < A[j]$
- Subject to the above condition the total increase in price is as large as possible, i.e. $A[j] - A[i]$ is greater than or equal to $A[k] - A[\ell]$ for any k, ℓ where the price only increases between k and ℓ.

Note that we are looking for the largest increase in **value**, not necessarily the sequence with the most elements. You will return the magnitude of the increase.

For example if $\mathtt{A} = [0, 10, 8, 15, 18, 17, 37]$ the correct answer is 20 corresponding to the last two elements of the array.

We will give full credit for any algorithm that runs in $\mathcal{O}(n \log n)$ time, but there are faster algorithms.

(a) Describe how you will solve the problem (in English or pseudocode). You do not have to prove your algorithm correct or analyze its running time in this part.

(b) Briefly explain why your algorithm works. (Our explanation is 2 sentences, depending on your algorithm, you might need more or less explanation).

(c) What is the running time of your algorithm? Briefly justify (1-2 sentences).

4.2. Are You Ready For Some Football?

We saw in class how to tell if the Mariners can make the playoffs, but what about the Seahawks?

An NFL game can end with one team winning and the other team losing, or with the teams tying. If a game ends in a tie, each team gets $1/2$ of a win.

You will be given a list of teams, with their current number of wins (which will be either an integer, or an integer plus $1/2$) along with a list of remaining games to be played. Your goal is to tell whether the Seahawks can finish with **more** wins than any other team (in class, we allowed for another team to have the same number of wins, in this problem the Seahawks need more to count as winning).

For example, if Team A has $1/2$ wins, the Seahawks have $1/2$ wins, and Teams B and C have 0 wins each, and the remaining games to be played are A vs. B, A vs. C, B vs. C, and B vs. the Seahawks, then it is impossible for the Seahawks to finish the mini-season with more wins than any other team.
You should return `False`.

If the current standings are Teams A, B, C, and the Seahawks all have 5 wins and the remaining games are A vs. B and C vs. the Seahawks, the Seahawks can still win (by winning their game, and having A vs. B end in a tie).
You should return `True`

(a) Describe how you will solve the problem (in English and/or pseudocode)

(b) Briefly explain why your algorithm works. Be sure to mention how it checks for the possibility of ties (our explanation is 3 sentences, depending on your algorithm, you might need more or less explanation).

(c) What is the running time of your algorithm? Briefly justify (1-2 sentences).

The Mystery section ends here

5. Survey

In the (hopefully unlikely) event that we have to offer 417 remotely again, we wanted to get feedback on the exam format so we can do better next time.

Please answer these questions to help us:

- How long did you spend on the exam (approximate to the nearest hour)?
- How long did you study for the exam before the exam started (approximate to the nearest hour)?
- How many other students did you talk to (please state their names as well)?
- If you talked to others, was it helpful?
- In how many of the four sections did you spend significant time attempting both problems?
- Did you like selecting between problems, or would you rather have just gotten one question per topic?

Celebration

You did it! Please enjoy this celebratory puppy photo.