Write your answers in the separate answer booklet.

You have 120 minutes (after you get the answer booklet) to answer five questions.

Please return this question sheet and your cheat sheet with your answers.

1. (a) Solve the following recurrences:
   - $A(n) = 2 \cdot A(n/2) + O(n^2)$
   - $B(n) = B(n/4) + 2 \cdot B(n/16) + O(\sqrt{n})$
   - $C(n) = 4 \cdot C(n/3) + O(n)$

   (b) Another member of your homework team has come up with the following recurrence to solve a dynamic programming problem whose input is an array $A[1..n, 1..n]$:

$$Foo(i) = \max \left\{ \left(\sum_{k=1}^{j} A[j,k] \right)^2 + Foo(j) \;\middle|\; 1 \le j < i \right\}$$

   The recurrence uses an implicit base case $\max \varnothing = 0$. You need to compute $Foo(n)$.

   Describe an appropriate memoization structure and evaluation order for this recurrence, and state the running time of the resulting iterative algorithm to compute $Foo(n)$.

2. Sham-Poobanana University has been closed due to damage caused by a nationwide infestation of rabid squirrels. You decide to complete your computer science degree at Sham-Poobanana's long-time rival Burgoo Polytechnic ("Go Boilers!"), which has somehow survived the squirrel plague.

   Burgoo has an extremely rigid CS program. There is a single set of classes that *all* CS majors *must* take, and prerequisites are strictly enforced. Burgoo does not offer proficiency exams or accept transfer credit. On the other hand, Burgoo's classes are extremely easy; they do not enforce class attendance; and they do not limit the number of classes you can take in a single term. You rationally decide to speed-run your degree, but how?

   (a) Suppose you are given a directed acyclic graph $G = (V, E)$, where every vertex represents a required class and each edge $u \to v$ indicates that class u must be completed before you can begin class v. Describe an algorithm that computes a class schedule that uses the minimum number of terms; for each required class, the output of your algorithm should specify the term when you will take that class.

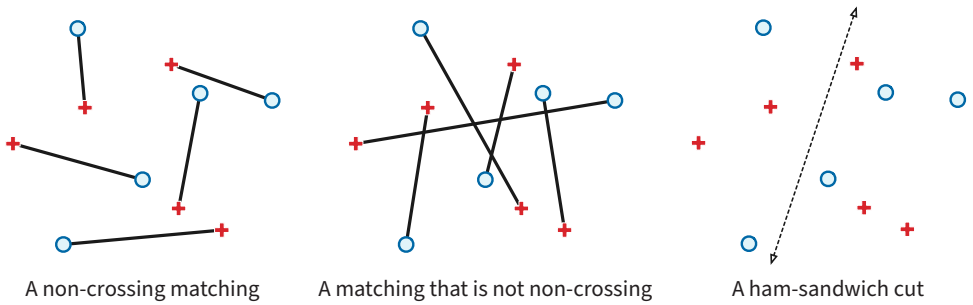
   (b) Burgoo relaxes its degree requirements due to massive overcrowding. Describe an algorithm that computes the minimum number of terms required to take ***at least 75%*** of the classes represented in G, while still obeying all prerequisite constraints. *[Hint: Use the output of your algorithm for part (a).]*

3. Let R be a set of n red points and let B be a set of n blue points in the plane. A ***non-crossing matching*** of R and B is a collection M of n line segments, each with one red endpoint and one blue endpoint, no two of which intersect. (M is called a *matching* because it *matches* each red point to a unique blue point.) A ***ham-sandwich cut*** for R and B is a line ℓ that satisfies the following conditions:

   - No point in $R \cup B$ lies directly on ℓ.
   - Exactly $\lfloor n/2 \rfloor$ red points and exactly $\lfloor n/2 \rfloor$ blue points lie above ℓ.

   Under some mild technical conditions, R and B must have at least one non-crossing matching and at least one ham-sandwich cut. (Don't try to prove that; in particular, don't worry about the technical conditions. Just trust me, bro.)

   The following figure shows two matchings (one of which is non-crossing) and a ham-sandwich cut for the same five red points (crosses) and five blue points (circles).



   Timothy has helpfully implemented a pair of subroutines for you.

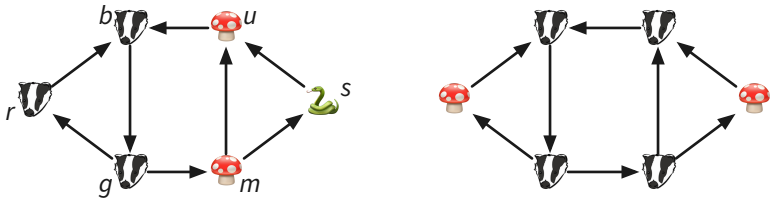
   - HamSammy(R, B) computes a ham-sandwich cut ℓ for two given n-point sets R and B, in $O(n)$ time.
   - Above?(p, ℓ) returns True if point p lies above line ℓ and False otherwise, in $O(1)$ time.

   Describe and analyze an algorithm to compute a non-crossing matching of two given n-point sets R and B, using Timothy's subroutines as black boxes. (Do *not* try to implement these subroutines yourself!)

4. Jonti "Mr. Weebl" Picking is making yet another badger video, this time using an audio sequencer. The sequencer is hard-wired with a directed graph $G = (V, E)$, in which each vertex produces a sound. Specifically, some vertices produce the spoken word "badger", others produce the spoken word "mushroom", and still others produce the spoken word "snake". To program the sequencer, Jonti needs to enter a closed walk in G; the sequencer then repeatedly traverses this walk, emitting the sound for each vertex that it visits.

   Jonti defines a closed walk in G to be ***orthodox***[1] if it has length 13, starts at a "badger" vertex, visits exactly ten more "badger" vertices followed by exactly two "mushroom" vertices, and then returns to the original "badger" vertex. Describe and analyze an algorithm that correctly determines whether G contains an orthodox closed walk.

   For example, if G is the graph on the left below, your algorithm should return TRUE, because the closed walk $b \to g \to r \to b \to g \to r \to b \to g \to r \to b \to g \to m \to u \to b$ is orthodox. However, if G is the graph on the right, your algorithm should return FALSE, because no walk in G visits two mushrooms in a row.



5. You are running a consulting company with two offices, one in San Francisco and the other in New York. Each week you must work in one of these two offices. Your analysts have given you two arrays $NY[1..n]$ and $SF[1..n]$ containing the profits you can earn at each office for each of the next n weeks. You'd prefer to work each week in city with higher profit, but there's a catch: Flying between New York and San Francisco costs \$1000.

   Your task is to design a travel schedule for the next n weeks that yields the maximum *total* profit. Your travel schedule can start at either city, and it can end at either city.

   For example, suppose you are given the following profit schedule:

SF	\$800	\$200	\$500	\$400	\$1200
NY	\$300	\$900	\$700	\$2000	\$200

   If you spend the first week in San Francisco, the next three weeks in New York, and the last week in San Francisco, your total profit for those five weeks is \$800 **− \$1000** + \$900 + \$700 + \$2000 **− \$1000** + \$1200 = \$3600.

   (a) There is an obvious greedy strategy: Each week, work in the city with higher profit. ***Prove*** that this greedy strategy does ***not*** always yield the maximum total profit.

   (b) Describe and analyze an algorithm to compute the maximum total profit you can earn.

[1] from the Greek word *orthos* (correct, regular) and the German word *Dachs* (badger)