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:
   - $A(n) = 3 \cdot A(n/4) + O(n)$
   - $B(n) = 3 \cdot B(n/2) + 4 \cdot B(n/4) + O(n^2)$
   - $C(n) = 2 \cdot C(n/2) + O(\sqrt{n})$

   (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 $X[1\,..\,n, 1\,..\,n]$.

$$Argle(i,\ell) = \sum_{j=i+1}^{\ell-1} \sum_{k=j+1}^{\ell-1} \max\left\{Argle(i,j),\ X[j,k],\ Argle(k,\ell)\right\}$$

   This recurrence uses an implicit base case $\sum\varnothing = 0$.

   Describe an appropriate memoization structure and evaluation order for this recurrence, and state the running time of the resulting iterative algorithm to compute $Argle(1,n)$.

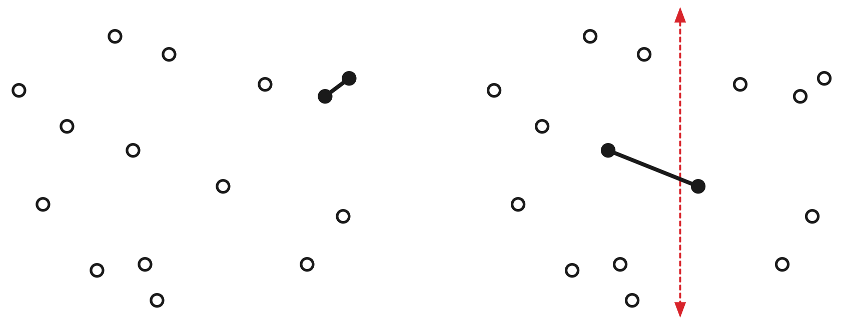
2. Suppose you are given a directed acyclic graph G whose nodes represent jobs that you can potentially perform. Each edge $u \rightarrow v$ in G indicates that you must complete job u *immediately before* starting job v. Thus, the jobs you complete must lie along a single directed path in G, and you must complete every job on that path in order. For each vertex v, you are also given two positive integers:
   - $v.deadline$ is the deadline for completing job v.
   - $v.reward$ is the reward for completing job v on time. (If you complete a job after its deadline, you will not receive a reward for that job, but you can still receive rewards for later jobs.)

   Each job requires exactly one hour to complete. Describe an algorithm to compute the maximum total reward you can earn, starting at time 0.

3. This problem asks you to design an algorithm to find the closest pair of points in a given set P of n points in the plane. Points in P are represented by an array $P[1..n]$ of coordinate pairs; the coordinates of the ith point are $(P[i].x, P[i].y)$. The points are sorted by their x-coordinates, and all x- and y-coordinates are distinct. In particular, we have $P[i].x < P[i+1].x$ for every index i. Timothy has helpfully implemented a pair of subroutines for you.

   - IWalkTheLine(P, a) computes the closest pair of points in a given n-point set P that cross the vertical line $x = a$, in $O(n)$ time. Specifically, this function returns the indices i and j of the closest pair of points such that $P[i].x \le a < P[j].x$. See the figure below for an example.
   - Distance(p, q) returns the Euclidean distance between two points p and q, in $O(1)$ time.

   Describe and analyze an algorithm to compute the Euclidean distance between the closest pair of points in P, using Timothy's subroutines as black boxes. (Do *not* try to implement Timothy's subroutines yourself!)



The closest pair of points (computed by your algorithm) The closest pair crossing a vertical line (computed by IWalkTheLine)

4. Suppose you are given an array $A[1..n]$ of integers. Describe and analyze an algorithm that computes the length of the longest increasing subsequence of A whose elements sum to a multiple of 67.

   For example, given the input array

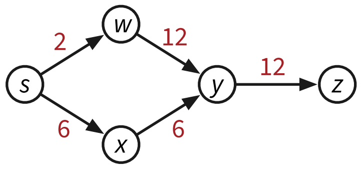
$$[1, 2, \underline{6}, \underline{42}, 3, 4, \underline{124}, 5, \underline{128}, 6, 7, \underline{173}, 8, \underline{225}, 9, \underline{374}, 10, 11],$$

   your algorithm should return the integer 7, which is the length of the increase subsequence $[6, 42, 124, 126, 173, 225, 374]$, whose elements sum to $1072 = 16 \cdot 67$. (The increasing subsequence $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ is longer, but its elements sum to 66, which is not a multiple of 67.)

   [Hint: Think about how you would solve the problem if we had asked for a multiple of 3 instead of a multiple of 67.]

5. Dijkstra Airlines is running a discount program to attract new customers. Their network of flights is modeled by a directed graph $G = (V, E)$ whose vertices represent airports and whose edges represent direct flights; each edge has a weight, which is the cost of the corresponding flight. While the discount program is in effect, if you purchase a trip consisting of one or more connected flights, you automatically get a 50% discount on *the most expensive direct flight* in that trip.

   Consider the ***partial*** flight network shown below. The cheapest flight from s to x is the direct flight $s \to x$, which costs $6/2 = 3$; the cheapest flight from s to y is $s \to w \to y$, which costs $2 + 12/2 = 8$; and the cheapest flight from s to z is $s \to x \to y \to z$, which costs $6 + 6 + 12/2 = 18$.



   Describe and analyze an algorithm to compute the cheapest trip from one airport to another. The input to your algorithm consists of the graph G, the edge weights (flight costs), the starting vertex/airport s, and the target vertex/airport t. Assume that it is possible to fly from any airport to any other airport (but not necessarily directly).