CS140 Final Exam

December 15, 2001

This is a 24 hours take-home exam. The exam is due by 6 PM on Saturday, December 22. This is a closed book/closed notes exam. You may use a one page cheat sheet to help you. You may not discuss the exam with anyone except me.

Your answers will be judged for clarity, conciseness, and correctness. You have more than adequate time to write your answers carefully. Illegible, confusing, and/or incomplete ideas will be judged harshly. The space provided should be adequate for your solutions.

Your exam should have 17 pages. Make sure it is complete before you begin. Good luck!

Name: ______________________________

Start Day/Time ______________________________

End Day/Time ______________________________

Part A: The first part of the exam consists of 25 short-answer questions. Each is worth 1 point. The space provided should be adequate for your solutions.

1. Rank the following functions by increasing rate of growth:
$$n, \;\; 3^{\lg n}, \;\; \lg n^{\sqrt{n}}, \;\; (\sqrt{n})^{\lg n}$$

2. Which of the following functions are $\Omega(n)$?
$$\lg(2n)^n, \quad \lg(3^n), \quad \lg^2(\sqrt{n}), \quad \lg(\sqrt{n^3}), \quad \ln(n^{\ln n})$$

3. Which of the following functions are $\omega(n)$?
$$\lg(2n)^n, \quad \lg(3^n), \quad \lg^2(\sqrt{n}), \quad \lg(\sqrt{n^3}), \quad \ln(n^{\ln n})$$

4. Give a tight (i.e. Θ) bound for the following recurrence relation:
$$T(n) = 3T(n/4) + cn^2$$

5. Give an example of a monotonically increasing function on the natural numbers, $f(n)$, such that $f(2n) \neq O(f(n))$.

6. Give an example of a monotonically increasing function on the natural numbers, $f(n)$, such $f(n^2) = O(f(n))$.

7. Is the following statement true or false? If for every n there is some input of size n for which algorithm A takes $3n^2$ steps then A is $\Omega(n^2)$.

8. Is the following statement true or false? If for every n there is some input of size n for which Algorithm A takes $3n^2$ steps then A is $O(n^2)$.

9. Is the following statement true or false? If $f(n)$ and $g(n)$ are monotonically increasing functions on the natural numbers then $f(n) = O(g(n))$ or $g(n) = O(f(n))$.

10. What does it mean to be a comparison-based sorting algorithm?

11. What is the minimum number of comparisons needed to find the minimum in a set of n integers? (Your answer should be exact.)

12. Give an example of a classic sorting algorithm that is $\theta(n \lg n)$.

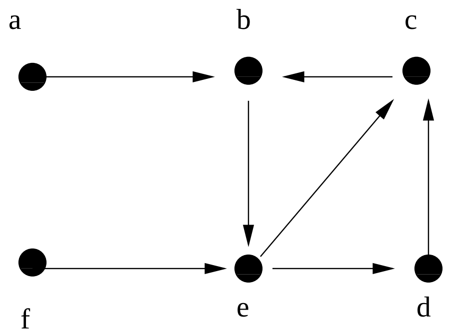
13. What property do the keys in a binary search tree satisfy?

14. What is the "heap" property?

15. Give a θ-bound on the time it takes to find the median of n integers.

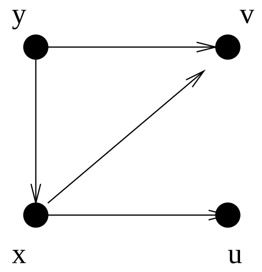
16. What is a "cut" in a graph?

17. Draw the DFS tree for the following digraph when alphabetical priority is used.



18. Give the timestamp table for the DFS search in the previous problem.

19. List the vertices of the following digraph in topological order.



20. What is a feasible flow in a flow network?

21. Give a yes-instance and a no-instance of 3SAT.

22. Prove that Vertex Cover is in NP.

23. What does it mean for a problem to be NP-complete? (There are two conditions you should mention.)

Part B: The second part of the exam consists of three problems. Each problem is worth 25 points.

1. In class we showed that any comparison-based sorting algorithm is $\Omega(n \lg n)$. This bound does not necessarily hold if the input is guaranteed to be "partially sorted." Consider the problem of sorting a list of distinct integers, $S : s_1, s_2, \ldots, s_n$, when the list is guaranteed to have at most k *inversions.* An inversion occurs at s_i $(1 < i < n)$ if s_i is either a peak or a valley; i.e. s_i is larger than both s_{i-1} and s_{i+1} or it smaller than both s_{i-1} and s_{i+1}. For example, the list $1, 2, 5, 4, -1, 10, 8$ has three inversions; it has peaks at 5 and 10 and a valley at -1.

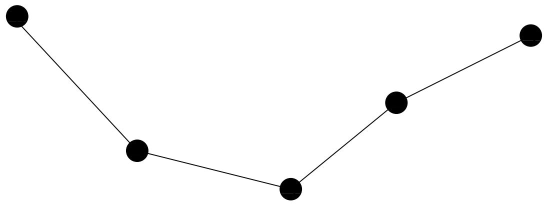
   For each of the following specifications of k (the maximum number of inversions) state whether the restricted problem can be solved in $o(n \lg n)$ or if the $\Omega(n \lg n)$ bound still holds. In either case prove your answer. (No credit will be given for an answer without a proof.)

   (a) (6 points) $k = 2$

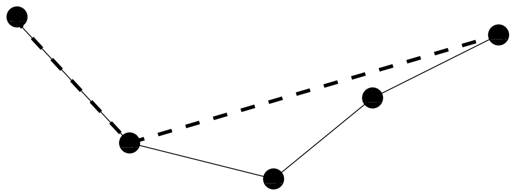
(b) (12 points) $k = \lg n$

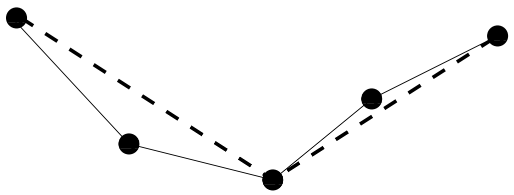
(c) (7 points) $k = n/2$

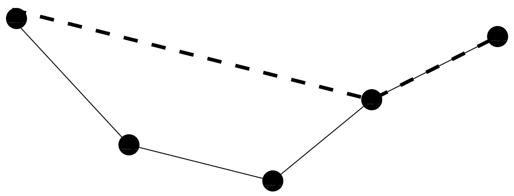
2. A poly-line is a sequence of points in 2-dimensional space connected by line segments. Following is an example of a poly-line on 5 points.



A poly-line L on n points may be approximated by a poly-line on m points ($n \geq m \geq 2$) by choosing the endpoints of L plus an additional $m - 2$ interior points. For example, the poly-line above can be approximated by a poly-line on 3 points in several ways as shown below. (The approximations are shown as dotted lines.)

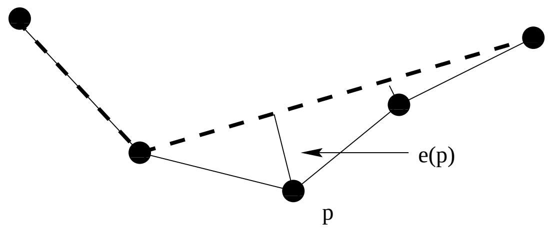


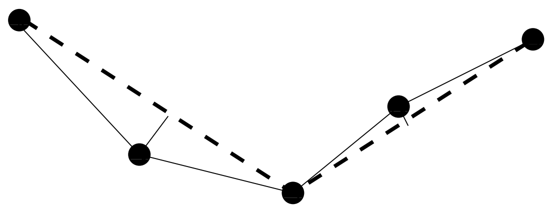


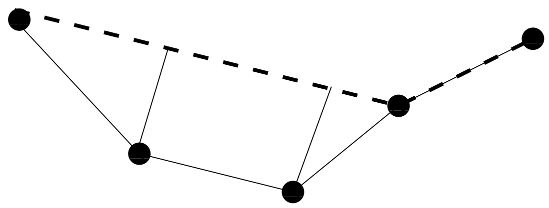


As you can see from the previous example, the choice of the intermediate points can dramatically affect the quality of the approximation. We will define the error of an approximation L' to the poly-line L as $\sum_{p \in L} e(p)$, where $e(p)$, the error of point p, is defined as follows. If p is a point in L' then $e(p) = 0$. If p is not a point of L' then $e(p)$ is the shortest distance from p to the line segment of L' that shortcuts p.

The errors for the earlier approximations are illustrated below.







In this problem we want to design an efficient algorithm that takes as input a poly-line L and an integer m and outputs an approximation L' that uses m points and has the smallest possible error. You can assume that $m \leq n$, where n denotes the number of points of L. You also may assume that you are given a subroutine *dist* that takes three points, p_1, p_2, p_3, and returns the shortest distance between p_1 and the line segment $\overline{p_2, p_3}$.

(a) (12 points) Suppose you have an algorithm $\mathcal{A}_{n-1}$ that will solve an instance L, m provided L has no more than $n-1$ points. Show how to use this algorithm to build one that works for poly-lines on n vertices. You may write your algorithm in pseudo-code but you also must give a clear, concise explanation of your algorithm in English. Be sure to carefully define any notation you use in your explanation. Assuming that the subroutine *dist* takes constant time, what is the running time of your recursive algorithm?

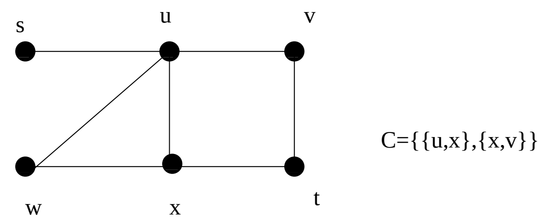
(b) (13 points) Design an efficient dynamic programming algorithm for this problem. As before, include an explanation of your algorithm in English. What is the running time of your algorithm?

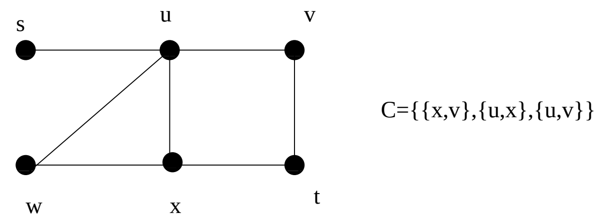
3. The Path with Forbidden Pairs (PFP) problem is defined as follows:

Input: A graph G containing vertices s and t and a collection C of pairs of vertices from G.

Question: Is there a path from s to t in G that visits at most one vertex in every pair in C.

(a) (2 point) State whether each of the following instances is a yes-instance or a no-instance of PFP. In the case of a yes-instance give an example of an appropriate s to t path.





(b) (2 points) In a search variant of the problem we want to find the actual path when one exists. Give a polynomial-time reduction from this search problem to the decision problem.

(c) Now show that PFP is NP-complete by giving a polynomial-time reduction from one of the NP-complete we've used in class. Specifically

i. (5 points) State which problem you'll use.

ii. (8 points) Describe your reduction.

iii. (1 point) Show the reduction can be done in polynomial time.

iv. (7 points) Prove the reduction works.