Algorithms – CMSC-27200
http://alg15.cs.uchicago.edu
Homework set #3 due January 28, 2015

Read the homework instructions on the website. The instructions that follow here are only an incomplete summary.

Hand in your solutions to problems marked "HW." Do not hand in problems marked "DO." If you hand in homework marked **"XC" (extra credit)**, do so on **separate and separately stapled sheets, please**. PRINT YOUR NAME ON EVERY SHEET you submit. We asked you to **use LaTeX to typeset your solutions;** starting with this homework, this is **required.** Hand in your solutions on paper, do not email.

When writing pseudocode, **explain the meaning of your variables.** Use comments to explain what is happening in each line. Also, give a short explanation of the idea behind the algorithm. **Describe all algorithms in this problem set in pseudocode. Elegance of your code matters.**

Carefully study the policy (stated on the website) on collaboration, internet use, and academic integrity.

In this problem set, every digraph is given in an *adjacency-list* representation, unless expressly stated otherwise. Recall that an adjacency-list representation consists of an array of vertices, where vertex u is the start of the linked list $\mathrm{Adj}[u]$ that lists all the out-neighbors of u in some order. An algorithm on a digraph $G = (V, E)$ is said to run in ***linear time*** if the number of steps is $O(|V| + |E|)$.
By "graph" we mean an undirected graph without self-loops and without parallel edges. A graph is a digraph satisfying the condition that for all $u, v \in V$, if $(u, v) \in E$ then $(v, u) \in E$ and $(u, u) \notin E$.

3.1 DO **(Applications of DFS)** Study the elegant and sometimes magical linear-time algorithms built on DFS. Read Cormen et al.

(a) Chap. 22-1: find various structural elements of graphs (articulation points, bridges, biconnected components)

(b) Chap. 22-4: topological sort

(c) Chap. 22-5: strongly connected components of a digraph

3.2 DO: review discrete probability, random variables, expected value from Discrete Mathematics, especially the solved "examples" after the "Linearity of expectation" section in Rosen's chapter on the expected value (Chap. 6.4 in the 6th edition).

3.3 **HW (3+8 points) (Membership cards)** The Jolly Spirits Club of Northern Central Wyoming has 2000 members. The Club decided to give every member a membership card and to celebrate the event by serving a glass of vodka to each member. The cards are numbered 1 to 2000. Each member receives a randomly drawn card.

(a) State the size of the sample space (number of possible outcomes) of this experiment.

(b) A *lucky member* is a member whose card number agrees with his or her year of birth. Lucky members receive a small prize. Determine the expected number of lucky members.

Half the credit for part (b) goes for the clear definition of the random variables you introduce. Explain the role of vodka in this problem. (Note: the method used in the solution is important for the analysis of randomized algorithms.)

3.4 DO: Let a_n and b_n be sequences of non-zero real numbers. We defined the relation $a_n \gtrsim b_n$ ("greater than or asymptotically equal") to mean $b_n \sim \min\{a_n, b_n\}$.

(a) Prove: $a_n \gtrsim b_n$ if and only if $a_n \sim \max\{a_n, b_n\}$.

(b) Prove: if $a_n, b_n > 0$ then $a_n \gtrsim b_n$ if and only if for every positive ϵ we have $a_n \geq (1-\epsilon)b_n$ for all sufficiently large n, i.e.,

$$(\forall \epsilon > 0)(\exists n_\epsilon)(\forall n \geq n_\epsilon)(a_n \geq (1-\epsilon)b_n).$$

3.5 **HW (8 points) (Sort-of-sorting)** Discount Algorithmics, Inc. (DA) sells algorithms with various performance guarantees. Their "Sort-of-Sorting" (SOS) algorithm comes with the rather modest guarantee that it will correctly sort at least 1% of the $n!$ permutations of any given list of n distinct real numbers. We are advised that SOS is comparison-based. DA advertises that *"We save on comparisons and pass the savings on to our customers."* Prove that this is a lie: if the guarantee is true then SOS must make $\gtrsim n \lg n$ comparisons on some inputs. (Note: the phrase "on some inputs" is synonymous with "in the worst case.") You may use the preceding exercise (about the $\gtrsim$ relation) without proof.

3.6 DO **(Big-Oh versus induction)** Joe claims he can prove that $2^n = O(1)$.

His proof goes by induction on n.
Base case: $2^1 = 2 = O(1)$.
Inductive step: Assume now that $2^{n-1} = O(1)$ (Inductive Hypothesis). Then $2^n = 2 \cdot 2^{n-1} = 2 \cdot O(1) = O(1)$. (In the last equation, Joe uses the fact that $2f(n) = O(f(n))$ for any function $f(n)$.) Done.

What's wrong with Joe's "proof"?

3.7 DO **(Bernoulli trials)** Recall from Discrete Mathematics: a *Bernoulli trial* is an experiment with two possible outcomes: *success* (S) and *failure* (F). Let p denote the probability of success; we assume $0 < p \leq 1$. Let X denote the number of times we need to repeat the experiment until the first "success" occurs. (We include the first success in the count. So if the sequence of outcomes is FFFS then $X = 4$.) Prove: $E(X) = 1/p$. (This result plays a role in the analysis of randomized algorithms.)

3.8 **HW (3 points) (Information Theory lower bound)** In class we stated that to identify one out of N objects one must make at least $\lg n$ binary (Yes/No) queries; this is the Information Theory lower bound for binary queries. (Recall: we use lg to denote base-2 logarithms.) State and prove the Information Theory lower bound for ternary queries (each query has 3 possible answers).

3.9 **(Fake coin algorithms)**

(a) DO: Out of 12 given coins we know that exactly one is fake but we don't know whether it is heavier or lighter. Given a balance, use three measurements to find the fake coin and decide whether it is heavier or lighter than the valid coins. (All the valid coins have the same weight. You can put any number of coins in each tray of the balance. Each measurement on the balance results in one of three possible outcomes: left-heavy, right-heavy, or equal.) Indicate why your procedure is correct.

(b) **XC (3 points)** Make your procedure *non-adaptive*: you need to tell in advance which coins to put in each tray in each of the measurements.

3.10 **(Fake coin lower bounds)**

(a) **HW (3 points)** Prove: if we have 14 coins in part (a) of the preceding problem ("Fake coin algorithms") then three measure-

ments do not suffice. Make your proof *very* short and elegant (just a couple of lines).

(b) **XC (5 points)** Prove: if we have 13 coins in part (a) of the preceding problem ("Fake coin algorithms") then three measurements do not suffice. Make your proof short and elegant (couple of short paragraphs).

Note: A solution to (b) will NOT earn you the 3 points for part (a). You need to give the even shorter proof of part (a) separately.

3.11 **XC (8 points, due February 4) (Counting fake coins)** We have n coins, at least one of which is fake. All the valid coins have the same weight; all the fake coins have the same weight; the fake coins are lighter. Find the number of fake coins using $O((\log n)^2)$ measurements on a balance.

Note: it is an **open problem** whether or not this could be solved with $o((\log n)^2)$ or perhaps even with just $O(\log n)$ measurements.

3.12 DO **(Divide-and-Conquer recurrence)**

(a) Study Strassen's fast matrix multiplication algorithm in the textbook.

(b) Study the handouts "Karatsuba's algorithm" and "The method of reverse inequalities."

(c) Strassen's algorithm leads to the recurrenct inequality $T(n) \leq 7T(n/2) + O(n^2)$. Use the method of reverse inequalities to prove that $T(n) = O(n^\beta)$ where $\beta = \log_2 7 \approx 2.81$.

3.13 **(Car race problem)** Let R be a subset of the $(n+1)^2$ points in the plane with integer coordinates between 0 and n. We call R the "race track." One of the points of R is designated as the start (S), another as the goal (G).

The points are represented as vectors (i, j). Cars are particles sitting on a point at any time. In one unit of time, a car can move from a point of R to another point of R, say from (i_1, j_1) to (i_2, j_2). The *speed vector* of the car during this time unit is defined as the vector $(i_2 - i_1, j_2 - j_1)$.

The *acceleration/deceleration* of the car is limited by the following constraint: from any one time unit to the next one, each coordinate of the speed vector can change by at most one.

For instance, if during time unit 6 the car was moving from point $(10, 13)$ to point $(16, 12)$ then its speed vector was $(6, -1)$ during this move; during the next time unit, the following are its possible speed vectors and corresponding destinations:

speed during time unit 7	location at the end of time unit 7
$(7, 0)$	$(23, 12)$
$(7, -1)$	$(23, 11)$
$(7, -2)$	$(23, 10)$
$(6, 0)$	$(22, 12)$
$(6, -1)$	$(22, 11)$
$(6, -2)$	$(22, 10)$
$(5, 0)$	$(21, 12)$
$(5, -1)$	$(21, 11)$
$(5, -2)$	$(21, 10)$

Of course only those locations are legal which belong to R (the car cannot leave the race track).

During time unit 0, the car rests at Start with speed $(0, 0)$. The objective is to decide whether or not the Goal is reachable at all and if so, to reach it using the minimum number of time units.

(a) **XC (3 points)** Construct an example where the optimal route visits the same point 100 times (at different speeds).

(b) **HW (4 points)** Prove: the speed of the car is bounded by $O(\sqrt{n})$ at all times. Here we interpret the "speed" as the ℓ_1-norm of the speed vector, i. e., if the speed vector is (v_1, v_2) then the "speed" is $|v_1| + |v_2|$. Specify the constant you get to justify the big-Oh claim; the smaller the constant, the better. (Assume n is large.)

(c) **HW (7 points)** Assume $|R| \geq n$. Find an optimal route in $O(|R| \cdot n)$ time. Describe your solution in clear English statements. Pseudocode not required. Algorithms discussed and analysed in class can be used as subroutines. Prove that your algorithm runs within the time claimed. *Hint.* Use BFS. Your main task is to construct the digraph to which to apply BFS. Do not overlook the possibility stated in part (a).

(d) **XC (4 points)** Solve the problem in $O(|R| \cdot n)$ time and space assuming $|R| < n$. (Note that you are not permitted to use an array with more than $O(|R| \cdot n)$ cells because of the space constraint.)

3.14 DO **(Interval sum)** We are given an array $A[1 \ldots n]$ of real numbers. The sum of the interval $[i, j]$ is the quantity $S[i, j] := \sum_{k=i}^{j} A[k]$. Find the maximum interval sum

$$S_{\max} = \max_{1 \le i,j \le n} S[i, j].$$

Find this value in *linear* time (i. e., the number of operations should be $O(n)$). Describe your solution in elegant and simple pseudocode. (Note: you are not required to output the interval with the maximum sum, just the value of the maximum sum.) Observe the following convention:

Convention: If $j < i$, we say that the interval $[i, j]$ is *empty*; the sum of the empty interval is zero. Empty intervals are admitted in the problem. Therefore $S_{\max} \ge 0$ even if all the $A[i]$ are negative.

Instructions. Use dynamic programming. Bear in mind that in dynamic programming exercises, half the credit goes for the clear and simple definition of the array of problems you use (the "brain" of the solution). Do not confuse the "brain" of the solution with the "heart" of the solution (the recurrence). **Explain the meaning of your variables!** *Elegance* and *simplicity* count. Do not use notation specific to some programming language, just the generic pseudocode instructions appearing in handouts.