CS161: Data Structures and Algorithms Handout #20
Stanford University Tuesday, 29 February 2000

Homework #7: Amortized analysis, exploring graphs
Due Date: Tuesday, 7 March 2000 — **hard deadline**; late days cannot be used for this homework.

Reading: Chapters 18, 23.

Recall that *exercises* are for you to work out on your own; *problems* are to be handed in.

Exercise 7-1. Do Exercise 18.2–3 on page 363 of CLR.

Exercise 7-2. Do Exercise 18.3–3 on page 366 of CLR.

Exercise 7-3. Do Problem 23.1–5 on page 468 of CLR.

Exercise 7-4. Do Problem 23.3–4 on page 484 of CLR.

Problem 7-1. Amortized analysis for two stacks [50 points]

Suppose there are two stacks called A and B, manipulated by the following operations:

push-A(d): Pushes a datum d onto stack A. Real Cost = 1.

push-B(d): Pushes a datum d onto stack B. Real Cost = 1.

multi-pop-A(k): Removes $min(k, size(A))$ elements from stack A. Real Cost $= min(k, size(A))$.

multi-pop-B(k): Removes $min(k, size(B))$ elements from stack B. Real Cost $= min(k, size(B))$.

transfer(k): Repeatedly pops elements from stack A and pushes them onto stack B, until either k elements have been moved, or A is empty. Real Cost=number of elements moved. (Note that you can transfer *only* from A to B.)

(a) Give amortized costs to each operation using the accounting method. Using your amortized costs show an $O(n)$ worst case bound on the cost of n operations.

(b) Give a potential function such that the amortized cost of each of the operations is constant, and evaluate the constant for each of the operations.

(c) Using your potential function show an $O(n)$ worst case bound on the cost of n operations.

Problem 7-2. Bipartite graphs [50 points]

An undirected graph $G = (V, E)$ is called *bipartite* if the nodes can be partitioned into two subsets A and B in such a way that all edges go between A and B.

(a) Prove that a graph is bipartite iff it can be 2-colored, that is iff all nodes can be colored with two colors so that no two adjacent nodes have the same color.

(b) Give an efficient algorithm to decide whether a graph is bipartite. A by-product of your algorithm should be the partition of V into A and B.