BU CLA CS 530: Analysis of Algorithms
Solutions for Mid-Term Test (14 March 1996)

Problem 1 (15 points).

(a) TRUE.
The worst-case run time for randomized quicksort happens (and can happen) when the partition in each iteration is around the minimum (or maximum) element, resulting in the recurrence $T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$.

(b) FALSE.
The problem of finding an optimal order to multiply a chain of matrices is solved by *dynamic programming* (not a greedy algorithm).

(c) TRUE.

(d) FALSE.

(e) TRUE.
A matching corresponds to a flow in a flow network. The matching is maximum iff the corresponding flow is maximum, which is true iff there is an augmenting path in the residual graph. To set up the flow network and residual graph takes $\mathcal{O}(V+E)$ time. To check for an augmenting path, using DFS or BFS, takes $\mathcal{O}(V+E)$ time. The total time is therefore $\mathcal{O}(V+E)$.

Problem 2 (10 points).

See the solutions for Problem 5 in Problem Set 2, and Problem 4 in Problem Set 3.

Problem 3 (10 points).

First, sort $A[1\,..,n]$ in $\mathcal{O}(n \log n)$ time. Second, if the entries in the sorted array are $x_1, x_2, \ldots, x_n$, we pair x_1 with x_n, x_2 with x_{n-1}, x_3 with x_{n-2}, and so on. The total time for the two phases is $\mathcal{O}(n \log n) + \mathcal{O}(n) = \mathcal{O}(n \log n)$.

We have to prove this procedure is correct. Suppose otherwise and we will get a contradiction. Suppose there is an optimal partition, i.e. a partition that minimizes the maximum sum, which does *not* match the smallest integer x_1 with the largest x_n, but instead matches x_1 with x_i and x_n with x_j, for some $i, j \in \{2, \ldots, n-1\}$. If $x_i \leq x_j$, then

$$x_i = x_1 + k, \qquad x_j = x_i + k + \ell, \qquad x_n = x_i + k + \ell + m$$

for some $k, \ell, m \geq 1$, so that $x_1 + x_i = 2x_i + k$ and $x_j + x_n = 2x_i + 2k + 2\ell + m$. But in this case we can get a better bound by matching x_1 with x_j and x_n with x_i: $x_1 + x_j = 2x_i + k + \ell$ and

$x_i + x_n = 2x_i + 2k + \ell + m$ — which contradicts the optimality of the partition. On the other hand, if $x_j \leq x_i$, then

$$x_j = x_1 + k, \qquad x_i = x_i + k + \ell, \qquad x_n = x_i + k + \ell + m$$

for some $k, \ell, m \geq 1$, and $x_1 + x_i = 2x_i + k + \ell$ and $x_j + x_n = 2x_i + 2k + \ell + m$. But we can get a better bound by matching x_1 with x_n and x_i with x_j: $x_1 + x_n = 2x_1 + k + \ell + m$ and $x_i + x_j = 2x_1 + 2k + \ell$ — another contradiction.

Problem 4 (10 points).

Let $|V| = n$, $|E| = m$, and $V = \{v_1, v_2, \ldots, v_n\}$. Construct a new graph $G' = (V', E')$ where

$$V' = V \cup \{ s_1, s_2, \ldots, s_n \} \qquad \text{and} \qquad E' = E \cup \{ (s_i, v_i) \mid i = 1, 2, \ldots, n \}$$

In words, G' is obtained from G by adding n new vertices $s_1, \ldots, s_n$, and then connecting s_1 to v_1, s_2 to v_2, etc. G' has $2n$ vertices and $m + n$ edges. We also define a new weight function for G':

$$w'(v_i, v_j) = w_1(v_i, v_j) + w_2(v_j)$$

for every distinct $(v_i, v_j) \in E$. To complete the definition of w', we also set $w'(s_i, v_i) = w_2(v_i)$ for every $i = 1, 2, \ldots, n$. Now,

$$p = \langle v_i, v_{j_1}, v_{j_2}, \ldots, v_{j_k}, v_\ell \rangle$$

is a least-weight (i.e. shortest) path in G relative to the weight functions w_1 and w_2 from v_i to v_ℓ if and only if

$$p' = \langle s_i, v_i, v_{j_1}, v_{j_2}, \ldots, v_{j_k}, v_\ell \rangle$$

is a least-weight path in G' relative to the weight function w' from s_i to v_ℓ. It suffices therefore to run one of the standard algorithms for "single-source shortest paths" (e.g. the Dijkstra's algorithm or, if there are negative weights, the Bellman-Ford algorithm) on G', to find the shortest-paths from s_i to every other vertex — n times, once for each of $s_1, s_2, \ldots, s_n$. If we use Dijkstra's algorithm, which can be implemented to run on G' in $\mathcal{O}((2n \log 2n) + m + n) = \mathcal{O}((n \log n) + m)$ time (see [CLR], Section 25.2), then the total time of our algorithm is $\mathcal{O}((n^2 \log n) + nm)$.

Created: 96.03.14