Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

Summer semester 2000
Solutions 10
July 25, 2000

Efficient Algorithms and Data Structures II

Problem 1 (10 credits)

Consider the following game, called *The Game of Erase*. You are given a string of length u with arbitrary characters. The only legal moves you are allowed to make are as follows.

Type 1. Erase a single character from the string.

Type 2. Erase two characters from the string if they are identical and adjacent.

The goal of the game is to find a minimal sequence of moves that will erase the whole string. For example, if the input string is **AACBC**, then the following sequence of moves is minimal: **AACBC** $\rightarrow$ **CBC** $\rightarrow$ **CC** $\rightarrow \epsilon$, where ϵ denotes the empty string.

For each statement below, determine, with proof, whether it is true or false.

a. If, for a given string $S = s_1 \cdots s_u$ of length u, the length of a minimal sequence of moves is n, then there exists a minimal sequence M that erases S such that one of the following is true.

1. The first move of M erases s_1, and the other $n-1$ moves erase $s' = s_2 \cdots s_u$ using a minimal sequence for S'.
2. For some index ℓ, $1 < \ell \leq u$, $s_1 = s_\ell$, and for some i, $0 \leq i < n$, the first i moves of M erase the substring $S' = s_2 \cdots s_{\ell-1}$ using a minimal sequence for S', the $(i+1)$-st move of M erases s_1 and s_ℓ, and the remaining $n-i-1$ moves of M erase the substring $S'' = s_{\ell+1} \cdots s_u$ using a minimal sequence for S''.

b. For all i, if there is a minimal sequence M of n moves that deletes s_i alone, i.e., using **type 1** move above, then for any $1 \leq j \leq n$, there exists a minimal sequence of moves for which deleting s_i is the j-th move.

c. For all $1 \leq i \leq u$, if there is a minimal sequence M of n moves that deletes s_i alone, then there exists a minimal sequence of moves M' such that M' first deletes $s_1 \cdots s_{i-1}$, then deletes s_i, and then deletes $s_{i+1} \cdots s_u$.

d. Let $p_{i,j}(S)$ denote the length of a minimal sequence of moves that erases the substring $s_i \cdots s_j$. If $j < i$, $p_{i,j}$ is defined to be 0. Then,

$$p_{i,j}(S) = \min \{1 + p_{i+1,j}(S), \min_{1 < \ell \leq j,\ s_i = s_\ell} \{1 + p_{i+1,\ell-1}(S) + p_{\ell+1,j}(S)\}\}.$$

Solution.

Ad a. The statement is true. Let $M = (m_1, m_2, \ldots, m_n)$ be a minimal sequence of moves that erases the string S. The first character s_1 has to be deleted either alone or with some other character (which is identical to it).

Case (i): s_1 was deleted alone. Suppose, s_1 was deleted in move m_j of M. Then, $M' = (m_j, m_1, m_2, \ldots, m_{j-1}, m_{j+1}, \ldots, m_n)$ also completely erases S, and we have another sequence of minimal length n in which s_1 is deleted in the first move.

Case (ii): s_1 is deleted along with character s_ℓ for some $1 < \ell \leq u$. Say this was move m_i of M. Consider the strings $S_1 = s_2 s_3 \cdots s_{\ell-1}$ and $S_2 = s_{\ell+1} s_{\ell+2} \cdots s_u$ (note that $S = s_1 S_1 s_\ell S_2$). All characters in S_1 must have been deleted before move m_i (otherwise we couldn't have deleted s_1 together with s_ℓ). Also, none of the moves involved in erasing S_1 can have involved any character in S_2 (s_ℓ still "blocks"). Suppose the sequence of moves that deleted S_1 were $M_1 = (m_{i_1}, m_{i_2}, \ldots, m_{i_s})$ for $1 \leq i_1 < i_2 < \cdots < i_s < i$. Let the remaining moves in M not involving m_i be $M_2 = (m_{j_1}, m_{j_2}, \ldots, m_{j_t})$ for $1 \leq j_1 < j_2 < \cdots < j_t \leq n$ (note that $s + t + 1 = n$). By our earlier argument, moves in M_1 involve only characters in S_1. Hence, we notice that the sequence $M' = (m_{i_1}, m_{i_2}, \ldots, m_{i_s}, m_i, m_{j_1}, m_{j_2}, \ldots, m_{j_t})$ also completely erases S. This sequence of moves first erases S_1, then s_1 and s_ℓ together, and finally S_2.

Ad b. The statement is false. Consider $S = \mathbf{ABA}$. A minimal sequence of moves of length two is as follows: $\mathbf{ABA} \to \mathbf{AA} \to \epsilon$. Clearly there is no sequence of two moves erasing S in which $\mathbf{B}$ is deleted alone in the second move.

Ad c. The statement is false. For the same counterexample as in (b), we notice that there can be no sequence of two moves which first delete the first character, **A**, then **B**, and finally the last character, **A**.

Ad d. The statement is true. Consider the sequence of moves erasing $S_{i,j} = s_i \cdots s_j$. By (a), there exists a minimal sequence $M_{i,j}$ of moves of length $p_{i,j}(S)$ that erases $S_{i,j}$ which satisfies one of the following.

1. The first move of $M_{i,j}$ erases s_i, and the other $p_{i,j}(S) - 1$ moves erase $s_{i+1} \cdots s_j$ using an appropriate minimal sequence.

2. For some index ℓ, $i < \ell \leq j$, $s_i = s_\ell$, and for some i, $0 \leq i < p_{i,j}(S)$, the first t moves of $M_{i,j}$ erase the substring $s_{i+1} \cdots s_{\ell-1}$ using an appropriate minimal sequence, the $(t+1)$-st move of $M_{i,j}$ erases s_i and s_ℓ, and the remaining $p_{i,j}(S) - t - 1$ moves of $M_{i,j}$ erase $s_{\ell+1} \cdots s_j$.

As $M_{i,j}$ is a minimal sequence of moves we obtain the following recurrence.

$$p_{i,j}(S) = \min \{1 + p_{i+1,j}(S), \min_{1 < \ell \leq j,\ s_i = s_\ell} \{1 + p_{i+1,\ell-1}(S) + p_{\ell+1,j}(S)\}\}.$$

Problem 2 (10 credits)

Let $G = (V, E)$ be a weighted, undirected graph. The *3Max-Cut* problem consists of partitioning V into three pairwise disjoint subsets V_1, V_2, and V_3, called clusters, such that the total cost of edges between clusters, called cross edges, is maximized. Consider the following approximation algorithm. Does APPROX-3MAX-CUT have a ratio bound? Why?

```
APPROX-3MAX-CUT(G)
    V1, V2, V3 ← ∅
    cluster, sum ← 0
    for each u ∈ V do
        mincost ← ∞
        for i ← 1 to 3 do
            cost ← 0
            for each v ∈ V_i do
                if (u, v) ∈ E then cost ← cost+c(u, v) endif
            endfor
            sum ← sum + cost
            if cost < mincost then
                mincost ← cost
                k ← i
            endif
        endfor
        V_k ← V_k ∪ {u}
        cluster ← cluster + mincost
        cross ← sum − cluster
    endfor
    return cross
```

Solution.

Each time through the outer **for** loop, the value of `sum` is increased by at least three times the increase in `cluster`. Since `sum` and `cluster` are initialized to zero, we have

$$\texttt{sum} \geq 3 \cdot \texttt{cluster} \quad \Rightarrow \quad \frac{1}{3}\texttt{sum} \geq \texttt{cluster}.$$

APPROX-3MAX-CUT returns

$$\texttt{cross} = \texttt{sum} - \texttt{cluster} \geq \texttt{sum} - \frac{1}{3}\texttt{sum} = \frac{2}{3}\texttt{sum}.$$

Denote by C the result returned by APPROX-3MAX-CUT and by C^* the cut obtained by some exact algorithm. Since $C^* \leq \texttt{sum}$, we have

$$\frac{2}{3}C^* \leq \frac{2}{3}\texttt{sum} \leq C \leq C^* \quad \Rightarrow \quad \frac{C^*}{C} \leq \frac{3}{2}.$$

Problem 3 (10 credits)

Multiprocessor Scheduling: Consider the simplest version of the multiprocessor scheduling problem where we have n jobs $j_1, j_2, \ldots, j_n$, with corresponding runtimes $p_1, p_2, \ldots, p_n$. The jobs are to be scheduled on m identical machines (or processors) so as to minimize the overall finish time. Consider the *list-approx-algorithm* A, that considers the n jobs one by one, and assigns the current one to the machine that is the least loaded. The load of a machine is the total run-time of the jobs assigned to it.

a. Show that $\frac{A(I)}{A^*(I)} \leq 2 - \frac{1}{m}$, where A^* is the exact algorithm that gives the optimum solution.

b. Show that the bound is tight. In other words, find a problem instance for which $\frac{A(I)}{A^*(I)} = 2 - \frac{1}{m}$.

Solution.

Ad a. Let W be the sum of the processing time of all jobs, i.e., $W = \sum_{i=1}^{n} p_i$. Obviously, $A^*(I) \geq W/m$.
Denote by $A_i(I)$ the finish time of the i-th machine. Assume that the h-th machine is the one with the highest finish time (i.e., $A_h(I) = A(I)$), and let i denote the index of the last job assigned to this machine. Since job j_i was assigned to the least loaded machine, the finish time of any other machine is at least $A_h(I) - p_i$. This implies that $W \geq m \cdot (A_h(I) - p_i) + p_i$, and that the following bound on $A(I)$ holds.

$$A(I) = A_h(I) \leq \frac{W}{m} + \frac{(m-1)p_i}{m}.$$

Since $A^*(I) \geq W/m$ and $A^*(I) \geq p_i$, we have that

$$A(I) \leq \frac{W}{m} + \frac{(m-1)p_i}{m} \leq A^*(I) + \frac{m-1}{m} A^*(I) = \left(2 - \frac{1}{m}\right) A^*(I).$$

Ad b. Consider $m(m-1)$ jobs of length 1 and one job of length m. Obviously, an optimal schedule has length m. If the longest job is the last one to be scheduled, then the finish time is $(m-1) + m = 2m - 1$ (first, the $m(m-1)$ small jobs are evenly distributed, and then the last job comes "on top" of some machine). Hence,

$$\frac{A(I)}{A^*(I)} = \frac{2m-1}{m} = 2 - \frac{1}{m}.$$