CS21003 Algorithms I, Autumn 2013–14

Class test 1

Maximum marks: 20 | Time: 11-Sep-2013 | Duration: 1 hour

Roll no: ____________ **Name:** ____________________________

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

You are given an array A of $n \geqslant 2$ integers $a_0, a_1, a_2, \ldots, a_{n-1}$. Your task is to evaluate the Maxminusian difference $(a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1})$. For two operands, we have $a \ominus b = a - b$. Problems arise when there are more than two operands. We conventionally take the binary difference operator as being left-to-right associative. This gives a unique meaning to the unparenthesized expression: $a_0 - a_1 - a_2 - \cdots - a_{n-1} = (\cdots((a_0 - a_1) - a_2) - \cdots - a_{n-2}) - a_{n-1}$. In Maxminusia, the law of left-to-right associativity does not hold. The rule instead is to produce a parenthesization that leads to a value as large as possible. As an example, take $n = 3$. The only different parenthesizations of $a_0 \ominus a_1 \ominus a_2$ are $(a_0 \ominus a_1) \ominus a_2 = (a_0 - a_1) - a_2$ and $a_0 \ominus (a_1 \ominus a_2) = a_0 - (a_1 - a_2)$. If $a_0 = 7$, $a_1 = -2$ and $a_2 = 3$, these two parenthesizations give the values 6 and 12, respectively. Therefore, $7 \ominus (-2) \ominus 3 = 12$. In what follows, you only need to compute the value of $(a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1})$. You do not need to compute a parenthesization that gives this value.

1. In this part, take $n = 4$. There are exactly five different ways to parenthesize $a_0 \ominus a_1 \ominus a_2 \ominus a_3$. One of these parenthesizations is given and evaluated at $a_0 = 4$, $a_1 = -7$, $a_2 = -3$ and $a_3 = 5$. Supply the other four parenthesizations and evaluate the expressions for $(a_0, a_1, a_2, a_3) = (4, -7, -3, 5)$. **(2)**

Parenthesization 1: $a_0 - ((a_1 - a_2) - a_3)$ = 13

Parenthesization 2: $a_0 - (a_1 - (a_2 - a_3))$ = 3

Parenthesization 3: $(a_0 - a_1) - (a_2 - a_3)$ = 19

Parenthesization 4: $((a_0 - a_1) - a_2) - a_3$ = 9

Parenthesization 5: $(a_0 - (a_1 - a_2)) - a_3$ = 3

The largest of these values is the Maxminusian difference $a_0 \ominus a_1 \ominus a_2 \ominus a_3 = 4 \ominus (-7) \ominus (-3) \ominus 5$.

A way to solve the Maxminusian difference problem is to look at all possible explicit parenthesizations of $(a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1})$. We evaluate these parenthesized expressions with $\ominus$ replaced by ordinary $-$, since in a fully parenthesized expression, these two operators are the same. We finally take the maximum of these values. An efficient implementation of this strategy uses dynamic programming.

Maximizing the difference $P - Q$ means maximizing P and minimizing Q. Therefore, you should also solve the Minminusian difference problem in which the parenthesization is such that the final value is as small as possible. Use two two-dimensional arrays **`maxd`** and **`mind`**. The entries **`maxd[i][j]`** and **`mind[i][j]`** are meant to store respectively the Maxminusian and the Minminusian differences of $a_i, a_{i+1}, \ldots, a_j$.

2. Illustrate how you should (initialize and) populate the two arrays **`maxd`** and **`mind`**. **(8)**

Solution We populate the (i, j)-th entries in the increasing sequence of $t = j - i$ (we always take $i \leqslant j$). The initialization corresponds to $i = j$. In this case, there is only one interpretation of the difference, namely, a_i.

```
for (i=0; i<n; ++i) maxd[i][i] = mind[i][i] = A[i];
```

Solution (Continued from last page)

Next, we consider $t = j - i = 1, 2, 3, \ldots, n-1$ in that sequence.

```
for (t=1; t<n; ++t) {
   for (i=0,j=t; j<n; ++i,++j) {
      /* Compute maxd[i][j] and mind[i][j] simultaneously */
      max = MINUS_INFINITY; min = PLUS_INFINITY;
      for (k=i; k<j; ++k) {
         diff = maxd[i][k] - mind[k+1][j];
         if (diff >= max) { max = diff; maxidx = k; }
         diff = mind[i][k] - maxd[k+1][j];
         if (diff <= min) { min = diff; minidx = k; }
      }
      maxd[i][j] = max;
      mind[i][j] = min;
   }
}
```

In the above implementation, the array locations i, j with $i > j$ do not need to be initialized or computed. We never need them. A space-saving implementation can be made by letting t and i index respectively the rows and the columns of the arrays. In this implementation, each row can be allocated exactly the amount of memory that is needed.

3. What value should you finally return? **(1)**

`maxd[0][n-1]`

4. What is the running time of this dynamic-programming algorithm? **(1)**

$\Theta(n^3)$

5. Propose an $O(n)$-time algorithm to solve the Maxminusian difference problem. Assume that $n \geqslant 2$. (**Remark:** For obtaining half credit, solve the problem when $a_0, a_1, a_2, \ldots, a_{n-1}$ are all positive or non-negative. Properly justify the correctness of your algorithm in order to get any credit, half or full.) **(8)**

Solution Let us write the sum $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1}$ as $\pm a_0 \pm a_1 \pm a_2 \pm \cdots \pm a_{n-2} \pm a_{n-1}$ with each $\pm$ so chosen that the expression corresponds to an explicit parenthesization and is as large as possible. The first $\pm$ must be chosen as $+$ and the second as $-$, since there are only two possibilities at a_0, namely, $((\cdots(a_0 - a_1) - \cdots$ and $((\cdots(a_0 - ((\cdots(a_1 - \cdots$. In both the cases, the plus sign before a_0 and the minus sign before a_1 cannot be avoided. It turns out that we can choose the remaining $\pm$ signs in such a way that each of the last $n-2$ terms (involving $a_2, a_3, \ldots, a_{n-1}$) has a positive contribution. More precisely, we have the following claim.

Claim: For all $n \geqslant 2$, we have $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1} = a_0 - a_1 + |a_2| + |a_3| + \cdots + |a_{n-1}|$.

Proof We proceed by induction on n. For $n = 2$, the result is obvious. So suppose that the result holds for some $n \geqslant 2$. We need to show that $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1} \ominus a_n = a_0 - a_1 + |a_2| + |a_3| + \cdots + |a_{n-1}| + |a_n|$. Let $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1} = E_1 - E_2 = a_0 - a_1 + |a_2| + |a_3| + \cdots + |a_{n-1}|$ for some fully parenthesized expressions E_1 and E_2. The $-$ between E_1 and E_2 is the outermost difference in the parenthesization. Now, consider the explicitly parenthesized expression: $E_1 - (E_2 - a_n)$ if $a_n \geqslant 0$, or $(E_1 - E_2) - a_n$ if $a_n < 0$. This expression evaluates to $a_0 - a_1 + |a_2| + |a_3| + \cdots + |a_{n-1}| + |a_n|$ in both the cases. Since $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1} \ominus a_n$ cannot be larger than this quantity (see the argument before the claim), we have $a_0 \ominus a_1 \ominus a_2 \ominus \cdots \ominus a_{n-1} \ominus a_n = a_0 - a_1 + |a_2| + |a_3| + \cdots + |a_{n-1}| + |a_n|$. •

This gives us the following linear-time algorithm.

```
diff = A[0] - A[1];
for (i=2; i<n; ++i) diff += (A[i] >= 0) ? A[i] : -A[i];
return diff;
```

This is, in essence, a greedy algorithm, because at every step you add the maximum possible contribution by a_i without worrying at all whether this greedy choice can have a detrimental effect in the future.

And well, you must now *know* that the Minminusian difference of $a_0, a_1, a_2, \ldots, a_{n-1}$ is $a_0 - a_1 - |a_2| - |a_3| - \cdots - |a_{n-1}|$.