NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC1007 Data Structures and Algorithms **2022/23 Semester 2**

Solution 4: Analysis of Algorithm

School of Computer Science and Engineering *Nanyang Technological University*

Q1 The function subset() below takes two linked lists of integers and determines whether the first is a subset of the second. Give the worst-case running time of subset as a function of the lengths of the two lists. When will this worst case happen?

```
typedef struct _listnode{
  int item;
  struct _listnode *next;
} ListNode;

//Check whether integer X is an element of linked list Q
int element (int X, ListNode* Q)
{
  int found;  //Flag whether X has been found
  found = 0;
  while ( Q != NULL && !found) {
     found = Q->item == X;
     Q = Q->next;
  }
  return found;
}

// Check whether L is a subset of M
int subset (ListNode* L, ListNode* M)
{
  int success; // Flag whether L is a subset so far
  success = 1;
  while ( L != NULL && success) {
      success = element(L->item, M);
      L = L->next;
  }
  return success;
}
```

S1 Let $|L|$ and $|M|$ indicate the length of the linked lists, L and M, respectively. The worst-case running time of subset:

- the first $|L|-1$ elements of L from the last $|L|-1$ elements of M in reverse order.
- the last element of L not in M

$\therefore$ Total number of comparisons between elements of L and M

$$
\begin{aligned}
&= |M| + (|M| - 1) + (|M| - 2) + \ldots + (|M| - (|L| - 2)) + |M| \\
&= |L||M| - (1 + 2 + 3 + \ldots + (|L| - 2)) \\
&= |L||M| - \frac{(|L| - 2)}{2}(1 + |L| - 2) \\
&= |L||M| - \frac{(|L| - 2)(|L| - 1)}{2} \\
&= \Theta(|L||M|)
\end{aligned}
$$

Here we assume that $|L| < |M|$

Q2 Find the number of printf used in the following functions. Write down its time complexity in Θ notation in terms of N.

```
1  void Q2a (int N)
2  {
3      int j, k;
4      for (j=1; j<=N;j*=3)
5          for(k=1;k<=N; k*=2)
6              printf("SC1007\n");
7  }
```

```
1   void Q2b (int N)
2  {
3      int i;
4      if(N>0)
5      {
6          for(i=0;i<N;i++)
7              printf("SC1007\n");
8          Q2b(N-1);
9          Q2b(N-1);
10     }
11 }
```

S2a Let K denote the number of iterations for the inner loop and J denote the number of iteration for the outer loop.
For the inner loop, we have

$$
\begin{aligned}
2^{K-1} &\leq N < 2^K \\
(K-1) &\leq \log_2 N < K \\
K &\leq \log_2 N + 1 < K + 1 \\
K &= \lfloor \log_2 N + 1 \rfloor = \lfloor \log_2 N \rfloor + 1
\end{aligned}
$$

For the outer loop, we have

$$\begin{aligned} 3^{J-1} &\leq N < 3^J \\ (J-1) &\leq \log_3 N < J \\ J &\leq \log_3 N + 1 < J + 1 \\ J &= \lfloor \log_3 N + 1 \rfloor = \lfloor \log_3 N \rfloor + 1 \end{aligned}$$

$\therefore$ The number of printf is $JK = (\lfloor \log_3 N \rfloor + 1)(\lfloor \log_2 N \rfloor + 1) = \Theta((logN)^2)$

S2b Let W(N) denote the number of printf used in the function with problem size of N

$$\begin{aligned} W(N) &= 2W(N-1) + N \\ &= 2(2W(N-2) + (N-1)) + N \\ &= 2^2 W(N-2) + 2(N-1) + N \\ &= 2^2(2W(N-3) + (N-2)) + 2(N-1) + N \\ &= \cdots \\ &= 2^{N-1}(1) + 2^{N-2}(2) + \cdots + 2^3(N-3) + 2^2(N-2) + 2(N-1) + N \\ &= \sum_{t=0}^{n-1} 2^t(n-t) \\ &= 2^{N+1} - 2 - N \end{aligned}$$

$\therefore$ The number of printf is $\Theta(2^N)$

Q1 A sequence, x_1, x_2, …, x_n, is said to be cyclically sorted if the smallest number in the sequence is x_i for some i, and the sequence, x_i, x_{i+1}, …, x_n, x_1, x_2, …, x_{i-1} is sorted in increasing order. Design an algorithm to find the minimal element in the sequence in $\mathcal{O}(\log n)$ time. What is the worst-case scenario?

S1 Let us see the following examples of cyclically sorted sequence:

1	2	3	4	5	6	7	8
6	7	8	1	2	3	4	5

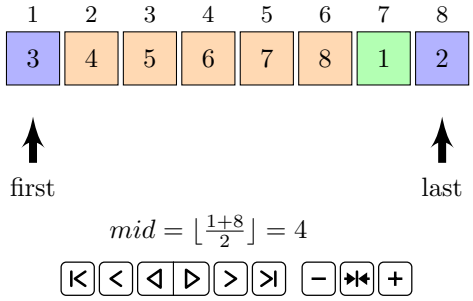
1	2	3	4	5	6	7	8
3	4	5	6	7	8	1	2

Given sequences above, x'_1, x'_2, …, x'_{j-1}, x'_j, x'_{j+1}, …, x'_{m-1}, x'_m, there exist an index, j that

$$x'_j < x'_{j+1} < \ldots < x'_m < x'_1 < \ldots < x'_{j-1}$$

How can we find out the index, j?
Let us select the middle element of the sequence, x'_{mid}. If the $x'_{mid} < x_m$ (the last element of the sequence), then the minimum definitely is in the first half (x'_1, x'_2, …, x'_{j-1}, x'_j) (including

itself). Otherwise, the minimum will be in the second half $(x'_{j+1}, \ldots, x'_{m-1}, x'_m)$.




```
1  int minimum(int array[], int n, int m) \\ n and m are the indices of the 1st
     and last elements respectively
2  {
3    int mid;
4    if (m == n) return array[n];  \\ this is the min
5    else {
6     mid=(n+m)/2;
7     if (array[mid] < array[m])        \\ middle < last
8        return minimum(n, mid); \\ find min in 1st half
9     else
10       return minimum(mid + 1, m); \\ find min in 2nd half
11   }
12   }
```

Algorithm of Q3

There is no difference. All cases have to run the same number of comparisons.