Deakin University

Data Structures and Algorithms

OnTrack Submission

Designing two simple algorithms

Submitted By:
Yuwei Wang
wangyuw
2019/09/05 15:48

Tutor:
Soudeh Kasiri bidhendi

September 5, 2019

Practical Task 5.2

1:

```
MERGE SORT(A, p, r)
if   p < r
    then q ← ⌊(p+r)/2⌋
        MERGE-SORT (A, p, q)
         MERGE-SORT (A, q+1, r)
         MERGE (A, p, q, r)

 MERGE(A, p, q, r)
     n1 ← q-p+1;
     n2 2 ← r-q;
     create arrays L[1..n1+1] and R[1..n2+1]
      for i ← 1 to n1
         do L[i] ← A[p + i-1]
        for j ← 1 to n2
        do R[j] ← A[q + j]
        L[n1+1] ← ∞
       R[n2+1] ← ∞
       i ← 1
       j ← 1
       for k ← p to r
       do if L[i] < R[j]
       then A[k] ← L[i]
       i ← i + 1
       else A[k] ← R[j]
       j ← j + 1
```

2:

Method:

1: Create two stack, one called stackData to store the current element, another one called stackMin to store the smallest element in every move.

2:To push, every time, when push one element into the stack, determine whether the stackMin is empty or not, if it is empty, push the element into the stack, if not, determine the two element size, if the first one is smaller or equal to the second one, push the first one data into the second one. If the second one is smaller than first one, push the number back to the stack

3:to pop, both stack element pop out the same time.

Pseudocode:

```
Push(int data)
{
    stackData.push
if(stackMin.empty())
{
        stackMin.push
}
Else{
     a = stackMin.pop()
     b = data >b ?a:data
     stackMin.push(b)
}
}

Pop()
{
Assert(!stackData.empty && !stackMin.empty());
stackData.pop();
stackMin.pop();
}
```