ESc 101: Fundamentals of Computing

Lecture 27

Mar 11, 2010

Outline

First Algorithm

```
sort(sequence S, size n)
{
    If (n == 1) return;
    Find minimum element of S;
    Insert it at the first location;
    Reset S to remaining sequence;
    sort(S, n-1);
}
```

The sort() function

```
// Sorts sequence of n numbers in S
void sort(int S[], int n)
{
    int i;
    if (n == 1) // sequence already sorted!
        return;
    i = find_minimum(S, n); // index of smallest element
    swap(S, S+i); // swap S[0] and S[i]
    sort(S+1, n-1);
}
```

Second Algorithm

```
merge_sort(sequence S, size n)
{
    If (n == 1) return;
    Split S into two halves S1 and S2;
    sort(S1, n/2);
    sort(S2, n/2);
    Merge S1 and S2 into a single sorted sequence and
        transfer to S;
}
```

The merge_sort() Function

```
// Sorts sequence of n numbers in S
void merge_sort(int S[], int n)
{
    int m;
    int T[N]; // temporary storage
    if (n == 1) // sequence already sorted!
        return;
    m = n / 2;
    merge_sort(S, m); // sort first m numbers
    merge_sort(S+m, n-m); // sort remaining numbers
    merge(S, m, S+m, n-m, T); // merge two sequences
    copy(T, n, S); // copy T to S
}
```

The merge() Function

```
/* Merge the two sorted sequences in S1 and S2, of
 * length n1 and n2 respectively, and store the result in T.
 */
void merge(int S1[], int n1, int S2[], int n2, int T[])
{
    int i; // index for T
    int j; // index for S1
    int k; // index for S2
```

The merge() Function

```
    for (i = 0, j = 0, k = 0; i < n1+n2; i++) {
        if (S1[j] < S2[k]) { // copy S1[j] in T
            T[i] = S1[j];
            j++;
        }
        else { // copy S2[k] in T
            T[i] = S2[k];
            k++;
        }
    }
}
```

An Algorithm for Merging

```
merge(sequence S1, size n1, sequence S2, size n2, sequence T)
{
   If (n1 == 0) copy S2 to T;
   If (n2 == 0) copy S1 to T;
   If (S1[0] < S2[0]) then {
      T[0] = S1[0];
      merge(remaining S1, n1-1, S2, n2, remaining T);
   }
   Else {
      T[0] = S2[0];
      merge(S1, n1, remaining S2, n2-1, remaining T);
}
```

The Function merge() Again

```
/* Merge the two sorted sequences in S1 and S2, of
 * length n1 and n2 respectively, and store the result in T.
 */
void merge(int S1[], int n1, int S2[], int n2, int T[])
{
    if (n1 == 0) { // S1 is empty
        copy(S2, n2, T); // copy S2 to T
        return;
    }
    if (n2 == 0) { // S2 is empty
        copy(S1, n1, T); // copy S1 to T
        return;
    }
```

The Function merge() Again

```
    if (S1[0] < S2[0]) { // S1[0] is  the smallest number
       T[0] = S1[0]; // copy the smallest number to T[0]
       merge(S1+1, n1-1, S2, n2, T+1); // merge the rest
    }
    else { // S2[0] is the smallest number
       T[0] = S2[0]; // copy the smallest number to T[0]
       merge(S1, n1, S2+1, n2-1, T+1); // merge the rest
    }
}
```