1 Workshop 3 - Induktion

I denne workshop kigger vi på MERGESORT, som er en sorteringsalgoritme. Algoritmen i pseudokode kan ses i Figur 1 eller i afsnit 5.4.4 i [Ros]. MERGESORT benytter MERGE, som kan ses i Figur 2 eller i afsnit 5.4.4 i [Ros]. Bemærk at den her er beskrevet på en lidt anden måde, men at metoden og tanken bag er den samme. Inden I begynder på selve opgaverne kan I med fordel læse of forstå afsnit 5.4.4. Som algoritmen er beskrevet herunder vil man starte med at kalde MERGESORT($L = (a_0, a_1, \ldots, a_{n-1})$, 0, $n-1$).

procedure MERGESORT($L = (a_0, a_1, \ldots, a_{n-1})$, l, r)
    if $l < r$ **then**
        $m = \left\lfloor \frac{r+l}{2} \right\rfloor$
        MERGESORT(L, l, m)
        MERGESORT($L, m+1, r$)
        $L =$ MERGE(L, l, m, r)
    return L

Figure 1: Mergesort

procedure MERGE(L, l, m, r)
    $L_1 = L[l, l+1, \ldots, m]$
    $L_2 = L[m+1, m+2, \ldots, r]$
    $i = 0$
    $j = 0$
    while $i < m - l + 1$ and $j < r - m$ **do**
        if $L_1[i] \leq L_2[j]$ **then**
            $L[l+i+j] = L_1[i]$
            $i = i + 1$
        else $L[l+i+j] = L_2[j]$
            $j = j + 1$
    if $i = m - l + 1$ **then**
        for $k = j \ldots r - m - 1$ **do**
            $L[l+i+k] = L_2[k]$
    else
        for $k = i \ldots m - l$ **do**
            $L[l+j+k] = L_1[k]$
    return L

Figure 2: Merge

1.1 Delopgave 1

1.1.1 Opgave

1. Implementer MERGE og MERGESORT.
2. Brug MERGESORT til at sortere listen $L = (5, 3, 8, 1, 6, 10, 7, 2, 4, 9)$

1.1.2 Løsning

1. Opgave 1 Taget fra https://www.geeksforgeeks.org/c-program-for-merge-sort/ fordi den fra moodle på ingen måde gad at virke uanset hvor meget jeg rettede i det. Den er alligevel baseret på næsten en direkt kopi af denne, så det går nok.

```
    #include<stdlib.h>
    #include<stdio.h>

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
```

```
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}
```

```c
void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
    int arr[] = { 5, 3, 8, 1, 6, 10, 7, 2, 4, 9 };
    int arr_size = sizeof(arr)/sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

2. Opgave 2

```
benjamin@DESKTOP-CNN41EU:~/c_opgaver$ ./mergekopi
Given array is
5 3 8 1 6 10 7 2 4 9
Sorted array is
1 2 3 4 5 6 7 8 9 10
```

1.2 Delopgave 2

1.2.1 Opgave

1. Bevis ved hjælp af induktion, at MERGESORT returnerer den sorterede liste hvis den får en liste med heltal som input. I beviset antages det, at MERGE returnerer en sammenflettet sorteret liste, hvis dellisterne $L[l, l+1, \ldots, m]$ og $L[m+1, m+2, \ldots, r]$ er sorteret.

1.2.2 Løsning

Lemma 1 i afsnit 5.4.4 i [Ros] siger, at to sorterede lister med m og n elementer kan sammenflettes (merged) til en sorteret liste ved højst $m+n-1$ sammenligninger.

1.3 Delopgave 3

1.3.1 Opgave

1. Antag at MERGE benytter $m+n-1$ sammenligninger til at sammenflette to lister med m og n elementer. Antag yderligere, at $n = 2^k$ og vis ved induktion over k (start med $k = 0$), at MERGESORT bruger præcis

$$n(\log_2(n) + 1) = 2^k(k + 1)$$

   sammenligninger hvis input-listen L har $n = 2^k$ elementer. Hvilket slags induktionsbevis brugte du?

2. Lav nu et induktionsbevis for, at MERGESORT bruger mindre end eller lig med $2n\log_2(n)$ sammenligninger for alle $n \geq 2$ under samme antagelser om MERGE som ovenfor.

   Hints til opgaven: Vi bliver nødt til at lave induktion over n. Bemærk at en liste med $n+1$ elementer deles op i to lister med $\left\lceil \frac{n+1}{2} \right\rceil$ og $\left\lfloor \frac{n+1}{2} \right\rfloor$ elementer i Mergesort. Desuden kan I få brug for følgende vurderinger:

   - $2\left\lfloor \frac{n+1}{2} \right\rfloor \log_2(\left\lfloor \frac{n+1}{2} \right\rfloor) \leq 2\left\lfloor \frac{n+1}{2} \right\rfloor \log_2(\left\lceil \frac{n+1}{2} \right\rceil)$
   - $\left\lfloor \frac{n+1}{2} \right\rfloor + \left\lceil \frac{n+1}{2} \right\rceil = \frac{n+1}{2}$
   - $\left\lceil \frac{n+1}{2} \right\rceil \leq \frac{2}{3}(n+1)$ når n er et positivt heltal
   - $\log_2(\frac{2}{3}(n+1)) = \log_2(n+1) - \log_2(\frac{3}{2})$
   - $n + 1 - 2(n+1)\log_2(\frac{3}{2}) < 0$ når n er positiv.

3. Hvilket slags induktionsbevis brugte du? Og hvad siger resultatet om tidskompleksiteten/store-O for mergesort?

1.3.2 Løsning