# *Lecture notes DAT037*

*Lukas Rahmn*

*2 november 2016*

## *Tidkomplexitet(tidsåtgång)*

1. Mäta impiriskt
2. noggrant ränkna anta intruktioner
3. förenklad modell/komplexitets analysi

$O(n^2), O(n \log n)$växer olika fort men $O(n^2)$ kan vara bättre algoritm för små n

### *Konstnadsmodeller*

Uniform modell

1. Tal kan vara hur stora som helst
2. Oändligt minne

Logaritmisk modell

1. Tidsåtgången för operationer på tal är propotionell mot antal bitar
2. Oändligt minne

$n = 2^k, k = \log n$

**Ändligt minne:** Algoritmen tar $O(n^2) \leq O(m^2), m = 4GB$ Alltså begränsar minnet mer än algoritmen.

### *Bästa-, <u>värsta</u>fallskomplexitet*

```
void filter(double[] x){
    for(i=0; i < x.length-5;i++){
        double s=0;
        for(int j=i; j < i+5; j++){
            s+=x[j];
        }
        x[i]=s/5;
    }
}
```

```
hasDuplicate(){
    for(...){
        if(a[i]==a[i+1]) return true;
    }
}
```

Bästa fall $O(1)$ , värsta $O(n)$

$T(n) = O(n)$

Notera att den inre loopen upprepas endast 5 gånger oavsett n, därför är den inre loopen konstant tid. Därför är totalt tidskomplexiteten $O(n)$ inte $O(n^2)$

```
void f(int[] x){
    for(int i=x.length; i >0; i /=2){
        x[i-1]=0;
    }
}
```

$i = 2^k$

$i = 2^{k-1}$

$\dots$

$i = 1 = 2^0$

$T(n) = O(logn)$

## *Dynamisk array*

Kom ihåg IntMultiSet. Den inehåller en int[] arr;

$T(2^k) = O(k+1) = O(k) = O(_2\log n) = O(logn)$

```
void add(int x){
    ...
}
```

P.ga av att arrayen är av fixed storlek så blir add $O(n)$

```
cclass IntMultiSet{
    int[] a;
    int size;
    void add(int x){
        if(size=a.length){
            int[] newa = new int[size+10];
            for(int i=0;i<size;i++){newa[i]=a[i];}
            a=newa;
        }
        a[size]=x;
        size++;
    }
}
```

Add fortfarande $O(n)$, värsta fall är fortfarande av linjär komplexitet

Ersätt istället size+10 med size*2.

### *Amorterad tidskomplexitet*

Add med

$[\,]$

$[x_1]$ ø o

$[x_1, x_2]$ ø ø o o o

$[x_1, x_2, x_3, \_]$ o o o o o

$[x_1, x_2, x_3, x_4]$ ø ø ø ø o o o

$[x_1, x_2, x_3, x_4, x_5, \_, \_, \_]$ o o o o o

Amorterad komplexitet $O(1)$.

## *Java generics*

```
void reverse(int[] arr)
    int tmp;
        for(...){
```

```
                tmp=arr[i];
                arr[i]=arr[i+1]
                    .
                }
```

Generisk

```
    void<E> reverse(E[] arr)
        E tmp;
            for(...){
                tmp=arr[i];
                arr[i]=arr[i+1]
                    .
                }
```

**Generics:**

1. metoder
2. interface
3. klasser

## *Datastrukturer*

### *ADT*

Man skiljer på vad man kan göra med en datastruktur och hur den är implementerad. Implementationerna kan skillja sig mellan olia listor men alla listor stödjer ett antal metoder.

### *Listor (Stackar,Köer)*

Listor (interface i java collection framework) ArrayList, LinkedList är exempel på konkreta listor.

| | |
|---|---|
| List | add(x),add(x,i),remove(i),get(i) |
| Stack | pop, push *LIFO* |
| Kö | enqueue,dequeue, *(FIFO)* |