```
void f1(int n)
{
  int t = sqrt(n);
  for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      // do something O(1)

    }
    n -= t;
  }
}
```

$t=\sqrt{n}$

$$\sum_{i=0}^{t}\sum_{j=0}^{n-1}\theta(1)=\sum_{i=0}^{\sqrt{n}}\theta(n)=\boxed{O(n^{3/2})}$$

Assume A is an array of size n+1.

```
void f2(int* A, int n)
{
  for(int i=1; i <= n; i++){
    for(int k=1; k <= n; k++){
      if( A[k] == i){
        for(int m=1; m <= n; m=m+m){
          // do something that takes O(1) time
          // Assume the contents of the A[] array are not changed
        }
      }
    }
  }
```

$$\sum_{i=1}^{n}\sum_{k=1}^{n}\left(\theta(1)+\sum_{m=1}^{i}\theta(1)\right)=$$

$$\sum_{k=1}^{n}\left(\theta(1)+\log(n)\right)=n+n\log n$$

$$\sum_{i=1}^{n} n+n\log n = n^2+n^2\log n$$

$$\boxed{O(n^2\log n)}$$

```
void f3(int* A, int n)
{
  if(n <= 1) return;
  else {
    f3(A, n-2);
    // do something that takes O(1) time
    f3(A, n-2);
  }
}
```

1. $T(n)=\theta(1)+T(n-2)+T(n-2)=\theta(1)+2T(n-2)$
2. $T(n)=\theta(1)+2\left(\theta(1)+T(n-4)+T(n-4)\right)=\theta(3)+4T(n-4)$
3. $T(n)=\theta(3)+4\left(\theta(1)+2T(n-6)\right)=\theta(7)+8T(n-6)$

$k=\frac{n-1}{2}$

$$T(n)=\theta(2^k-1)+2^kT(n-2k)$$

$$T(n)=\theta(2^{\frac{n-1}{2}}-1)+2^{\frac{n-1}{2}}(\theta(1))=2^{n-1/2}-1+2^{n-1/2}$$

$$\boxed{O(2^{\frac{n-1}{2}})}$$

```
int f (int n)
{
  int *a = new int [10];
  int size = 10;
  for (int i = 0; i < n; i ++)
     {
        if (i == size)
          {
             int newsize = 4*size;
             int *b = new int [newsize];
             for (int j = 0; j < size; j ++) b[j] = a[j];
             delete [] a;
             a = b;
             size = newsize;
          }
        a[i] = i*i;
     }
}
```