Analysis and Design of Algorithms

Sergio Carbone

April 6, 2019

1 Warm up

The main diffrence if we divide the array of numbers into 3 parts would be the base of the logarithm. The asymptotic running time instead of $nlog_2(n)$ would be $nlog_3(n)$.

2 Competitive programming

- **(100 - The 3m + 1 problem)**

```
#include <iostream>
using namespace std;

int to1(int n)
{
  int count=1;
    while(n!=1)
  {
    if (n%2!=0)
      {
      n=3*n+1;
      }else{
      n=n/2;
    }
    count++;
  }
  return count;
}

int main(int argc, char const *argv[])
{
  int max,a,b;
  while(cin>>a>>b)
  {
    max=0;
    if (a>b)
    {
      for (int i = b; i < a+1; ++i)
      {
        if (max<to1(i))
```

```
            {
              max=to1(i);
            }
          }
        }else{
          for (int i = a; i < b+1; ++i)
            {
              if (max<to1(i))
                {
                  max=to1(i);
                }
            }
          }
        cout<<a<<" "<<b<<" "<<max<<endl;
      }
      return 0;
    }
```

23111017	100 The 3n + 1 problem	Accepted	C++11	0.310	2019-04-04 18:46:02

- **(458 - The Decoder)**

```
#include <iostream>
using namespace std;
int main(int argc, char const *argv[])
{
  string sentence;
  while(getline(cin,sentence))
  {
    for (int i = 0; i < sentence.size(); ++i)
    {
      sentence[i]=sentence[i]-7;
    }
    cout<<sentence<<endl;
  }
  return 0;
}
```

23115503	458 The Decoder	Accepted	C++11	0.020	2019-04-05 15:37:04

3 Simulation

Merge Sort

```
#include <iostream>
using namespace std;

void merge(int arr[],int i,int m,int f)
  {
    int t1=m-i+1;
    int t2=f-m;
    int arr1[t1];
    int arr2[t2];
```

```
    for (int j = 0; j < t1; ++j)
    {
      arr1[j]=arr[i+j];
    }
    for (int k = 0; k < t2; ++k)
    {
      arr2[k]=arr[k+m+1];
    }

    int a=0,b=0,c=i;
    while(a<t1 && b<t2)
    {
      if (arr1[a]<=arr2[b])
      {
        arr[c]=arr1[a];
        a++;
      }else{
        arr[c]=arr2[b];
        b++;
      }
      c++;
    }
    while(a<t1)
    {
      arr[c]=arr1[a];
      a++;
      c++;
    }
    while(b<t2)
    {
      arr[c]=arr2[b];
      b++;
      c++;
    }

  }

void mergesort(int arr[], int i, int f)
{
  int m = (i+f)/2;
  if(i<f)
  {
    mergesort(arr,i,m);
    mergesort(arr,m+1,f);
    merge(arr, i,m, f);
  }


}


int main(int argc, char const *argv[])
{

  mergesort(array,0,7);
  for (int i = 0; i < 7; ++i)
```

```
  {
    cout<<array[i]<<" ";
  }
  return 0;
}
```

Insert Sort

```
#include <iostream>
using namespace std;

void insertsort(int arr[],int n)
{
  int key;

  for (int i = 1; i < n; ++i)
  {
    key = arr[i];
    int j = i-1;

    while(j>=0 && arr[j]>key)
    {
      arr[j+1]=arr[j];
      j--;
    }
    arr[j+1]=key;

  }
}


int main(int argc, char const *argv[])
{
  int array[7]={5,7,2,3,5,6,0};
  insertsort(array,7);
  for (int i = 0; i < 7; ++i)
  {
    cout<<array[i]<<" ";
  }

  return 0;
}
```

With the help of the next program we can confirm the effectiveness of the *Merge Sort* algorithm

```
#include <algorithm>
#include <chrono>
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
using namespace std::chrono;

void merge(int arr[],int i,int m,int f)
```

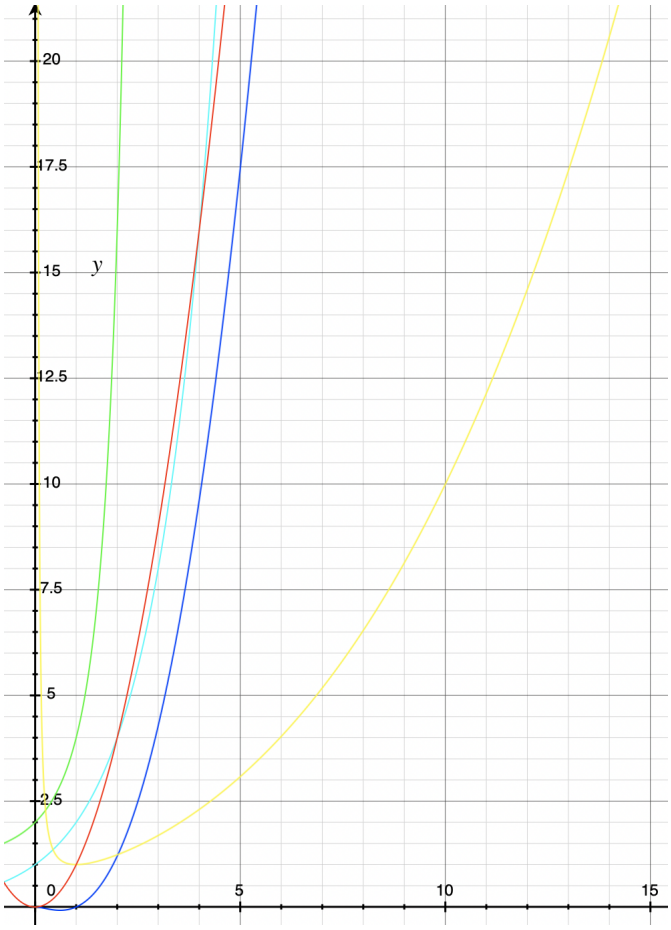
```
  {
    int t1=m-i+1;
    int t2=f-m;
    int arr1[t1];
    int arr2[t2];

    for (int j = 0; j < t1; ++j)
    {
      arr1[j]=arr[i+j];
    }
    for (int k = 0; k < t2; ++k)
    {
      arr2[k]=arr[k+m+1];
    }

    int a=0,b=0,c=i;
    while(a<t1 && b<t2)
    {
      if (arr1[a]<=arr2[b])
      {
        arr[c]=arr1[a];
        a++;
      }else{
        arr[c]=arr2[b];
        b++;
      }
      c++;
    }
    while(a<t1)
    {
      arr[c]=arr1[a];
      a++;
      c++;
    }
    while(b<t2)
    {
      arr[c]=arr2[b];
      b++;
      c++;
    }

  }

void mergesort(int arr[], int i, int f)
{
  int m = (i+f)/2;
  if(i<f)
  {
    mergesort(arr,i,m);
    mergesort(arr,m+1,f);
    merge(arr, i,m, f);
  }


}

void insertsort(int arr[],int n)
```

```
{
  int key;

  for (int i = 1; i < n; ++i)
  {
    key = arr[i];
    int j = i-1;

    while(j>=0 && arr[j]>key)
    {
      arr[j+1]=arr[j];
      j--;
    }
    arr[j+1]=key;

  }
}



int main(int argc, char const *argv[])
{


  srand (time(NULL));
    int random_number;
  random_number = rand() % 10000 + 1;
     int array1[random_number];
     int array2[random_number];
  int cant = random_number;



  for (int i = 0; i < cant; ++i)
  {
    random_number = rand() % 10000 + 1;
    array1[i]=random_number;
    array2[i]=random_number;

  }

  auto start1 = high_resolution_clock::now();

  insertsort(array1,cant);


  auto stop1 = high_resolution_clock::now();

  auto duration1 = duration_cast<microseconds>(stop1 - start1);

    cout << "Time taken by insertsort: "
         << duration1.count() << " microseconds" << endl;

  cout<<endl;

  //-------------
  auto start2 = high_resolution_clock::now();
```

```
    mergesort(array2,0,cant);

    auto stop2 = high_resolution_clock::now();

    auto duration2 = duration_cast<microseconds>(stop2 - start2);

      cout << "Time taken by mergesort: "
           << duration2.count() << " microseconds" << endl;

    cout<<endl;



      return 0;

}
```

4 Research

5 Wrapping up



1. $n^{log(n)} - >$Amarillo
2. $n^2 - >$Rojo
3. $n^2 log(n) - >$Azul
4. $2^n - >$ Celeste
5. $2^{2^n} - >$ Verde