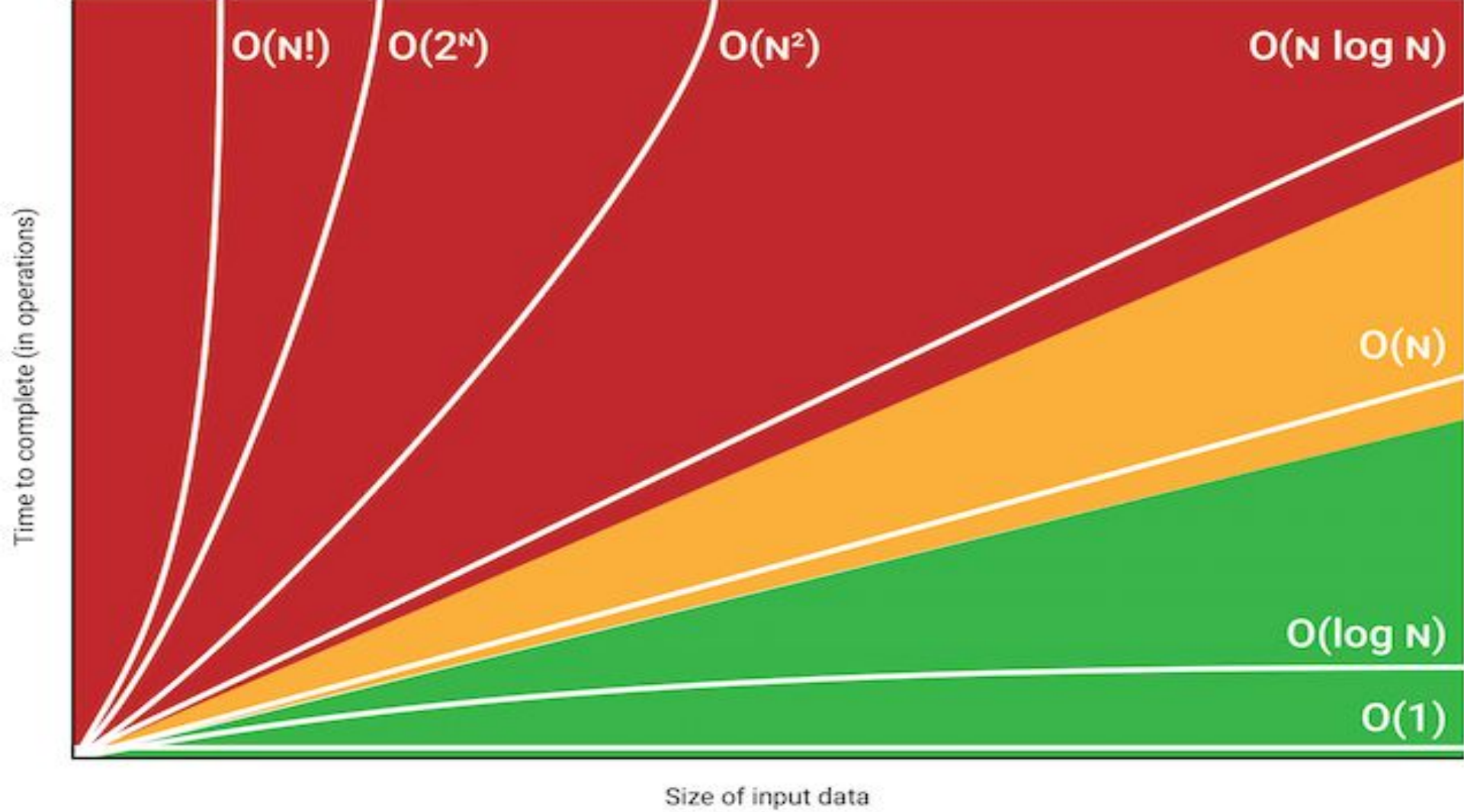
Clase 2: complejidad y sumas parciales

José & Jere


O(N!)
O(2^N)
O(N^2)
O(N log N)
O(N)
O(log N)
O(1)
Time to complete (in operations)
Size of input data

Problema: maximum subarray sum

Input: un arreglo A de largo n

Output: la máxima suma en un subarreglo

ejemplo:

A = [-5,10,1,-2,3]

Soluciones con distintas complejidades

A lo bruto es O(n**3)

Otra solución puede ser O(n**2)

Problema: obtener rápidamente sumas en rango.

input: un arreglo A de largo n, m queries de forma <l,r>

output: m números, las sumas en los rangos [l,r)

ejemplo:

A = [1,4,3,0,-7,10]

suma en [2, 5)

Solución: sumas parciales

Construyó el arreglo S de la forma

S[0] = 0;

Para i desde 0 hasta n: S[i+1] = S[i] + A[i];

[1,4,3,0,-7,10] -> [0,1,5,8,8,1,11]

Suma en [2,5) = s[5] - s[2] = 1 - 5 = -4

Estructuras útiles de c++: Vectores

```cpp
int main(){
  ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  // Declaro un vector
  vector<int> vect;
  // Leo su largo
  int n;
  cin>>n;
  int e;
  // Leo n elementos y los agrego al vector con push_back()
  for(int i = 0; i<n; ++i){
   cin>>e;
   vect.push_back(e);
  }
  // Accedo al primer elemento
  cout<<vect[0]<<'\n';
  // Pregunto el tamaño del vector
  cout<<vect.size()<<'\n';
  return 0;
}
```

Problemas:

https://cses.fi/problemset/task/1643

https://codeforces.com/problemset/problem/363/B

https://codeforces.com/problemset/problem/433/B

Solución Maximum Subarray Sum

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int main(){
  ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n;
  cin>>n;
  vector<ll> nums(n), psums(n+1);
  for(auto& i:nums) cin>>i;
  psums[0] = 0;
  for(int i=1; i<n+1;++i){
      psums[i] = psums[i-1] + nums[i-1];
  }
  ll res = nums[0], prevmin = 0;
  for(int i=1; i<n+1;++i){
      res = max(res,psums[i]-prevmin);
      prevmin = min(prevmin, psums[i]);
  }
  cout<<res<<'\n';
  return 0;
}
```

Solución Fence

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
  ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,k;
  cin>>n>>k;
  vector<int> fence(n),psums(n+1);
  for(auto& i:fence) cin>>i;
  psums[0] = 0;
  for(int i = 0; i<n ; ++i){
      psums[i+1] = psums[i] + fence[i];
  }
  int res = 0, cmp = psums[k];
  for(int i=k; i<n+1;++i){
      if(psums[i] - psums[i-k]<cmp){
          cmp = psums[i] - psums[i-k];
          res = i-k;
      }
  }
  cout<<++res<<'\n';
  return 0;
}
```

Solución KM's Stones

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    long long n;
    vector<long long> current;
    long long cur;
    cin >> n;
    long long i = 0;
    vector<long long> partial_unord;
    partial_unord.push_back(0);
    while(i<n){
        cin >> cur;
        current.push_back(cur);
        partial_unord.push_back(partial_unord[i] + current[i]);
        ++i;
    }
    sort(current.begin(),current.end());                    // Sort() ordena el vector en orden creciente
    vector<long long> partial_ord;
    partial_ord.push_back(0);
    i=0;
    while(i<n){
        partial_ord.push_back(partial_ord[i] + current[i]);
        ++i;
    }
    int m;
    cin >> m;
    int t;
    long long l;
    long long r;
    long long res;
    while(m--){
        cin >> t;
        cin >> l;
        cin >> r;
        if(t==1){
            res = partial_unord[r] - partial_unord[l-1];
        }
        else{
            res = partial_ord[r] - partial_ord[l-1];
        }
        cout<< res << endl;
    }
}
```