Assalammualaikum

Mohammad Hanif Furqan Aufa Putra

5025221161

Kuis PBO C

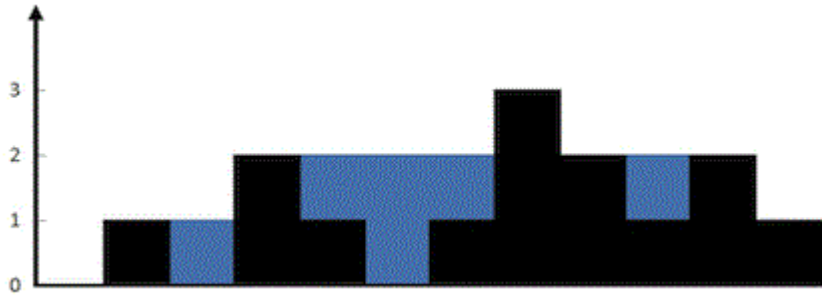
Trapping Rain Water

Time limit **1** second

Memory limit **128** MiB

Given nn non-negative integers representing an elevation map where the width of each bar is 11.

Compute how much water it can trap after raining.



Input data

The first line contains the value of n (n≤105)n (n≤105).

The second line contains nn non-negative integers h1,h2,...,hn (hi≤105)h1,h2,...,hn (hi≤105).

Output data

Print how much water can be trapped after raining.

Examples

Input example #1

```
12
0 1 0 2 1 0 1 3 2 1 2 1
```

Output example #1

```
6
```

Input example #2

```
6
4 2 0 3 2 5
```

Output example #2

```
9
```

Langkah-langkah untuk "Trapping Rain Water":

1. Observasi:

- Masalah "Trapping Rain Water" mengharuskan saya untuk menghitung berapa banyak air yang dapat terjebak di antara puncak-puncak dalam peta pada contoh. Untuk memahami konsep ini, kita perlu memperhatikan beberapa hal:

a. Air hanya bisa terperangkap jika ada dua puncak (bar) yang lebih tinggi yang membatasi area tertentu.

b. Air yang terperangkap diukur sebagai jumlah air yang diisi di antara dua puncak tersebut hingga tinggi air mencapai puncak terendah di antara keduanya.

- **Referensi:**

Konsep algoritma ini terinspirasi bagaimana air hujan dapat terperangkap di antara dua objek yang lebih tinggi seperti atap atau puncak-puncak bukit. Untuk mengukur berapa banyak air yang terperangkap, kita perlu memeriksa tinggi setiap puncak (bar) dalam konteks yang lebih luas.

2. Pemilihan Class STL:

Stack (Tumpukan):

Alasan: Saya menggunakan class stack dari STL (Standard Template Library) C++ karena masalah ini membutuhkan pendekatan yang mengingat ketentuan dari kuis dan harus mengakses elemen terbaru dengan cepat. Stack sangat cocok karena elemen terakhir yang dimasukkan akan selalu diperiksa terlebih dahulu. Ini mirip dengan bagaimana air terjebak di antara dua puncak, di mana puncak terakhir yang dilihat akan mempengaruhi apakah air dapat terjebak di antara mereka atau tidak.

Vector (Vektor):

Alasan: Saya menggunakan class vector untuk menyimpan tinggi masing-masing bar dalam peta elevasi. Vektor digunakan karena kita perlu mengakses tinggi setiap bar dengan mudah dan sejumlah variabel berurutan. Vektor memungkinkan akses dengan indeks dan pertambahan cepat dari elemen baru sesuai dengan input.

Tentu, mari bahas strategi algoritma, pemilihan STL, dan konsep ADT (Abstract Data Type) yang digunakan dalam menyelesaikan masalah "Trapping Rain Water" menggunakan C++ dengan STL.

1. Strategi Algoritma:

Observasi:

Masalah "Trapping Rain Water" mengharuskan kita untuk menghitung berapa banyak air yang dapat terjebak di antara puncak-puncak dalam peta elevasi. Untuk memahami konsep ini, kita perlu memperhatikan beberapa hal:

a. Air hanya bisa terperangkap jika ada dua puncak (bar) yang lebih tinggi yang membatasi area tertentu.

b. Air yang terperangkap diukur sebagai jumlah air yang diisi di antara dua puncak tersebut hingga tinggi air mencapai puncak terendah di antara keduanya.

Referensi Teori:

Konsep algoritma ini terinspirasi oleh bagaimana air hujan dapat terperangkap di antara dua objek yang lebih tinggi seperti atap atau puncak-puncak bukit. Untuk mengukur berapa banyak air yang terperangkap, kita perlu memeriksa tinggi setiap puncak (bar) dalam konteks yang lebih luas.

2. Pemilihan Class STL:

Stack (Tumpukan):

Alasan: Kami menggunakan class stack dari STL (Standard Template Library) C++ karena masalah ini membutuhkan pendekatan yang mengingat informasi sebelumnya dan harus mengakses elemen terbaru dengan cepat. Stack sangat cocok karena elemen terakhir yang dimasukkan akan selalu diperiksa terlebih dahulu. Ini mirip dengan bagaimana air terjebak di antara dua puncak, di mana puncak terakhir yang dilihat akan mempengaruhi apakah air dapat terjebak di antara mereka atau tidak.

Vector (Vektor):

Alasan: Kami menggunakan class vector untuk menyimpan tinggi masing-masing bar dalam peta elevasi. Vektor digunakan karena kita perlu mengakses tinggi setiap bar dengan mudah dan sejumlah variabel berurutan. Vektor memungkinkan akses dengan indeks dan pertambahan cepat dari elemen baru sesuai dengan input.

3. Abstract Data Type (ADT):

Stack (Tumpukan):

Penggunaan: Stack digunakan sebagai tumpukan indeks bar-bar yang mungkin akan berperan dalam menahan air.

Abstraksi: Stack adalah struktur data abstrak (ADT) yang mengikuti prinsip "Last In, First Out" (LIFO). Dalam konteks ini, stack digunakan untuk memantau bar-bar yang belum memiliki pasangan yang lebih tinggi, sehingga air dapat terjebak di antara mereka saat pasangan yang lebih tinggi ditemukan.

Vector (Vektor):

Penggunaan: Vector digunakan sebagai struktur data untuk menyimpan tinggi masing-masing bar dalam peta elevasi.

Abstraksi: Vector adalah ADT yang digunakan untuk menyimpan sejumlah elemen dengan indeks numerik yang berurutan. Dalam kasus ini, vektor digunakan untuk menyimpan data tinggi setiap bar dalam urutan yang sesuai dengan input.

4. Kesimpulan:

Strategi algoritma dalam masalah "Trapping Rain Water" adalah dengan menggunakan stack untuk melacak bar-bar yang belum memiliki pasangan yang lebih tinggi dan menghitung air yang terperangkap berdasarkan tinggi dan jarak antara dua pasang puncak yang lebih tinggi.

Pemilihan class STL seperti stack dan vector sangat relevan dengan masalah ini karena mereka menyediakan struktur data yang sesuai dengan kebutuhan algoritma dan abstraksi yang digunakan untuk memodelkan masalah ini.

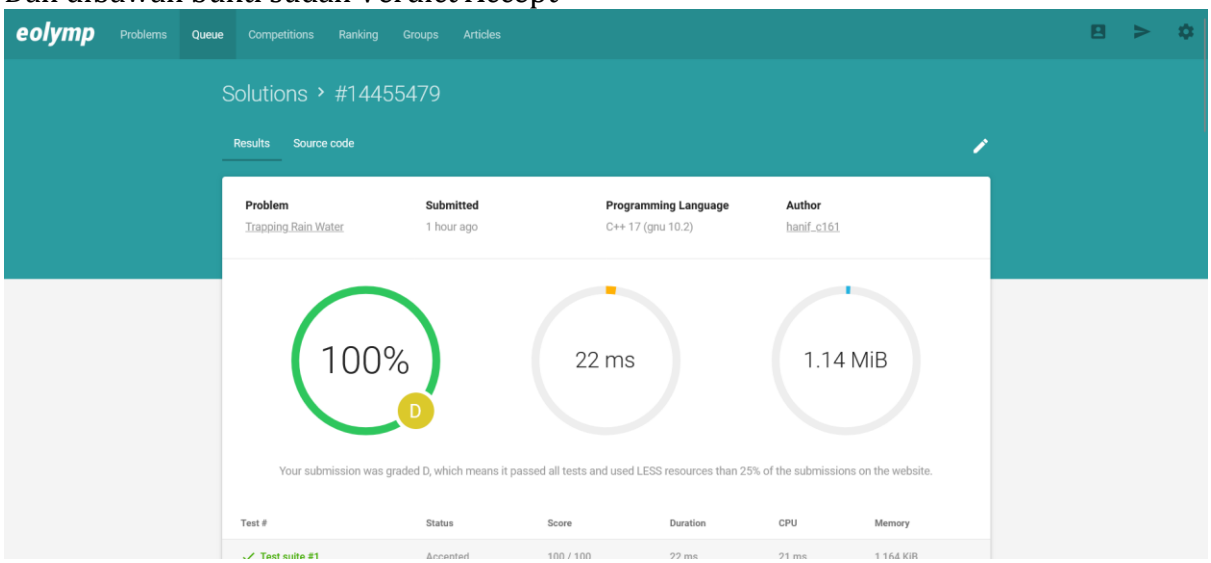
Dan dibawah ini merupakan keseluruhan code

```cpp
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

int trapRainWater(vector<int> &height)
{
    int n = height.size();
    if (n <= 2)
        return 0; // Tidak dapat menjebak air dengan kurang dari 3 bar

    stack<int> st; // Membuat tumpukan (stack) untuk menyimpan indeks bar
    int water = 0; // Variabel untuk menghitung air yang terperangkap

    for (int i = 0; i < n; ++i)
    {
        while (!st.empty() && height[i] > height[st.top()])
        {
            int topIdx = st.top();
            st.pop();

            if (st.empty())
                break; // Tidak ada batas kiri

            int distance = i - st.top() - 1; // Menghitung jarak antara dua bar
            int boundedHeight = min(height[i], height[st.top()]) - height[topIdx];
    // Menghitung tinggi air yang terperangkap
            water += distance * boundedHeight;
    // Menambahkan air yang terperangkap ke total air yang terperangkap
        }
        st.push(i); // Memasukkan indeks bar ke dalam stack
    }

    return water; // Mengembalikan jumlah total air yang terperangkap
}

int main()
{
    int n;
    cin >> n; // Membaca jumlah bar
    vector<int> elevationMap; // Vektor untuk menyimpan tinggi bar-bar

    for (int i = 0; i < n; ++i)
    {
        int height;
        cin >> height; // Membaca tinggi setiap bar dari input
        elevationMap.push_back(height);
    }

    int trappedWater = trapRainWater(elevationMap); // Menghitung air yang terperangkap
    cout << trappedWater << endl; // Menampilkan hasil jumlah air yang terperangkap ke layar

    return 0;
}

```

Dengan menjalan program tersebut diperoleh hasil seperti dibawah ini

```
                    > cd "f:\Coding\PBO-2023\Kuis\" ; if ($?) { g++ Trapping_Rain_Water.cpp -o Trapping_Rain_Water } ; if ($?) { .\Trapping_Rain_Water }
12
0 1 0 2 1 0 1 3 2 1 2 1
6
PS F:\Coding\PBO-2023\Kuis>
                    > cd "f:\Coding\PBO-2023\Kuis\" ; if ($?) { g++ Trapping_Rain_Water.cpp -o Trapping_Rain_Water } ; if ($?) { .\Trapping_Rain_Water }
6
4 2 0 3 2 5
9
PS F:\Coding\PBO-2023\Kuis>
```

Dan dibawah bukti sudah Verdict Accept



Terima Kasih telah membaca, Waalaikumsalam wr wb.