**ASSESSMENT FORM**

**Course:** **COMP6049001 - Algorithm Design and Analysis**

**Method of Assessment:** **Case Study**

**Semester/Academic Year : 3/2022-2023**

**VJUDGE PROFILE :** [**PROFIL**](https://vjudge.net/user/2501994222)

**Name of Lecturer : Surya Sujarwo**

**Date : 13 November 2022**

**Class : LB01 – COMP6049001**

**Topic : Review**

**Anggota Kelompok**

**1. Charles - 2501994222**

**2. jhonny martin - 2501995156**

**3. Syauqi Akmal Deffansyah - 2501992740**

**Penjelasan Solved Problem A Partly Medley**

1. User masukkan banyaknya inputan ( n ) dan maksimal perbedaan yang dibolehkan (th).

2. Scan inputan array sesuai dengan jumlah inputan (n) terlebih dahulu.

3. Kita menggunakan teknik brute force dengan 3 for loop karena soal meminta 3 exactly students. Namun di dalam 3 for loop harus mempunyai iterasi yang berbeda untuk menghindari kesalahan duplicate student lebih dari satu kali. Jika iterasi 3 for loop sudah berbeda semua kita baru menyelidiki apakah ketiga array loop iterasi mempunyai masing masing mempunyai perbedaan yang lebih sedikit dari th. Lakukan pengecekan pengkondisian apabila dari ketiga array mempunyai perbedaan masih dalam batas th, jika benar maka kita melakukan pengecekan max (nilai maksimal dari ketiganya) apakah lebih besar dari nilai max yang sekarang . Jika iya diganti dengan nilai yang baru, jika tidak tetap menggunakan nilai max dengan nilai yang terbesar sebelumnya dan kita tambahkan perhitungan kemungkinan perbedaan rating.

Sample Input#1

5 150

1400 1425 1250 4000 1300

**Iterasi i = 1, j = 2, z = 3**

i = 1400

j = 1425

z =1250

cek perbedaan ketiganya

abs(1400 – 1425) = 25 ( dalam batas 150)

abs(1400 – 1250) = 150 ( dalam batas 150)

abs(1425 – 1250) = 175 ( tidak dalam batas 150)

max = 0

count = 0

**Iterasi i = 1, j = 2, z = 4**

i = 1400

j = 1425

z = 4000

cek perbedaan ketiganya

abs(1400 – 1425) = 25 ( dalam batas 150)

abs(1400 – 4000) = 2600 ( tidak dalam batas 150)

abs(1425 – 4000) = 2575 ( tidak dalam batas 150)

max = 0

count = 0

**Iterasi i = 1, j = 2, z = 5**

i = 1400

j = 1425

z = 1300

cek perbedaan ketiganya

abs(1400 – 1425) = 25 ( dalam batas 150)

abs(1400 – 1300) = 100 (dalam batas 150)

abs(1425 – 1300) = 125 (dalam batas 150)

max = 4125

count = 1

**Iterasi i = 1, j = 3, z = 4**

i = 1400

j = 1250

z = 4000

cek perbedaan ketiganya

abs(1400 – 1250) = 150 ( dalam batas 150)

abs(1400 – 4000) = 2600 (tidak dalam batas 150)

abs(1250 – 4000) = 2750 (dalam batas 150)

max = 4125

count = 1

**Iterasi i = 1, j = 3, z = 5**

i = 1400

j = 1250

z = 1300

cek perbedaan ketiganya

abs(1400 – 1250) = 150 ( dalam batas 150)

abs(1400 – 1300) = 100 (dalam batas 150)

abs(1250 – 1300) = 50 (dalam batas 150)

max = 4125 ( 4125 > 3950)

count = 2

**Iterasi i = 1, j = 4, z = 5**

i = 1400

j = 4000

z = 1300

cek perbedaan ketiganya

abs(1400 – 4000) = 2600 (tidak dalam batas 150)

abs(1400 – 1300) = 100 (dalam batas 150)

abs(4000 – 1300) = 2700 (tidak dalam batas 150)

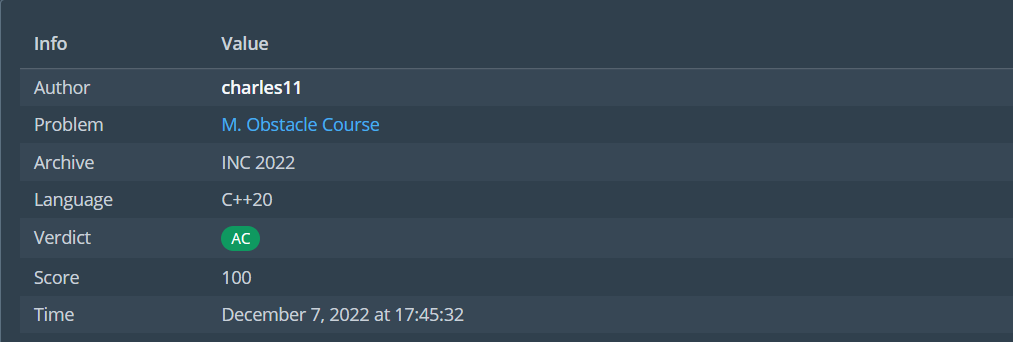
max = 4125

count = 2

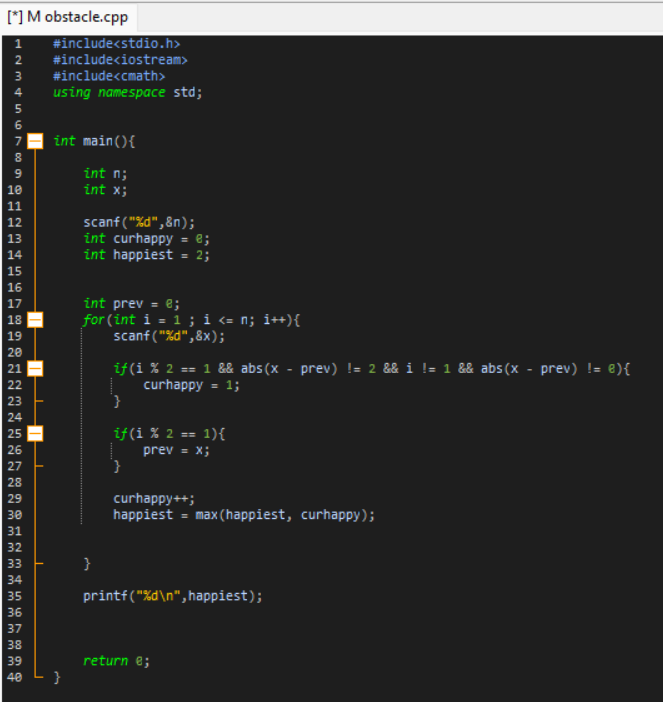
dst

sehingga kita mendapatkan nilai max = 4125 dengan probabilitas kemungkinan 3 student dibawah batas maksimal adalah 2

**#1 Unsolved Problem M Obstacle Course**

****

**Source Code :**

****

**Penjelasan Algoritma Unsolved Problem M (Obstacle Couse)**

1. User memasukkan inputan n (int)

2. Set currhappy adalah 0 dan happiest(kesenangan terbaik) = 2. Happiest terkecil minimal sudah pasti dua karena dari soal inputan minimal adalah 2 yang ganjil adalah ketinggian dan genap adalah 0 (bisa diatur sesuai keperluan)

Contoh : Inputan 1 0 maka bisa kita sesuai dengan mengganti 0 menjadi 2, agar mendapatkan happiest yang maksimal.

3. Iteration (for loop) segment sesuai dengan inputan user, lalu di setiap loop-nya karena kita mulai dari i = 1, maka ganjil adalah inputan selain 0 dan genap adalah 0 ( sesuaikan dengan keperluan ). Di setiap iterasinya maka apabila user memasukkan angka selain 0 ( daerah ganjil && bukan iterasi pertama) karena iterasi pertama tidak mempunyai inputan 0 sebelumnya dan kita harus selalu menyimpan inputan daerah ganjil untuk perbandingan inputan daerah ganjil selanjutnya.

Maka apabila banyak inputan data lebih dari dua maka disaat inputan i (ganjil) dimulai dari i = 3, 5, 7 dst. Maka akan masuk kedalam pengecekan pengkodisian (if) untuk menentukan apakah curhappy akan direset atau tidak dengan ketentuan apabila absolute(inputan daerah ganjil sebelumnya dikurangi inputan daerah ganjil terkini tidak sama dengan 2 dan tidak sama dengan 0) artinya 1 atau lebih dari dua maka kesenangan akan direset dimulai dari satu lagi. Karena apabila sama dengan dua kesenangan bisa berkelanjutan seperti

Contoh absolute(inputan daerah ganjil sebelumnya dikurangi inputan daerah ganjil terkini sama dengan 2) apabila hal ini didapatkan apabila user melakukan sample inputan seperti 1 0 3 , maka 0 bisa menjadi 2 agar dapat kesenangan dapat berkelanjutan(minimal berbeda satu setiap n).

Contoh absolute(inputan daerah ganjil sebelumnya dikurangi inputan daerah ganjil terkini sama dengan 0) hal ini didapatkan apabila user melakukan sample inputan seperti 1 0 1 , maka 0 bisa menjadi 2 agar dapat kesenangan dapat berkelanjutan (minimal berbeda satu setiap n).

4. Disetiap iterasi currhappy akan ditambah dan selalu dibandingkan dengan maksimal dari happiest sekarang dan currhappy agar happiest selalu menyimpangan kesenangan terbanyak.

Sample Input #1

9

6 0 4 0 8 0 12 0 14

**Variabel**

**Curhappy = 0;**

**Happiest = 2; (minimal sudah pasti dua)**

**Iterasi 1**

6 ( daerah ganjil ) simpan di dalam sebuah variabel untuk prev untuk perbandingan daerah ganjil selanjutnya.

Curhappy++

Curhappy = 1;

Happiest = 2;

**Iterasi 2**

0 ( daerah genap )

Curhappy++

Curhappy = 2;

Happiest = 2;

**Iterasi 3**

4 ( daerah ganjil bukan iterasi pertama) masuk perkondisian abs(4-6) = 2, maka currhappy dapat berkelanjutan

4 = simpan di dalam sebuah variabel untuk prev untuk perbandingan daerah ganjil selanjutnya.

Curhappy++

Curhappy = 3;

Happiest = 3;

**Iterasi 4**

0 ( daerah genap )

Curhappy++

Curhappy = 4;

Happiest = 4;

**Iterasi 5**

8 ( daerah ganjil bukan iterasi pertama) masuk perkondisian abs(8-4) != 2, maka currhappy direset menjadi satu

8 = simpan di dalam sebuah variabel untuk prev untuk perbandingan daerah ganjil selanjutnya.

Curhappy++

Curhappy = 2;

Happiest = 4;

**Iterasi 6**

0 ( daerah genap )

Curhappy++

Curhappy = 3;

Happiest = 4;

**Iterasi 7**

12 ( daerah ganjil bukan iterasi pertama) masuk perkondisian abs(12-8) != 2, maka currhappy direset menjadi satu

12 = simpan di dalam sebuah variabel untuk prev untuk perbandingan daerah ganjil selanjutnya.

Curhappy++

Curhappy = 2;

Happiest = 4;

**Iterasi 8**

0 ( daerah genap )

Curhappy++

Curhappy = 3;

Happiest = 4;

**Iterasi 9**

14 ( daerah ganjil bukan iterasi pertama) masuk perkondisian abs(14-2) == 2, maka currhappy tidak direset

14 = simpan di dalam sebuah variabel untuk prev untuk perbandingan daerah ganjil selanjutnya.

Curhappy++

Curhappy = 4;

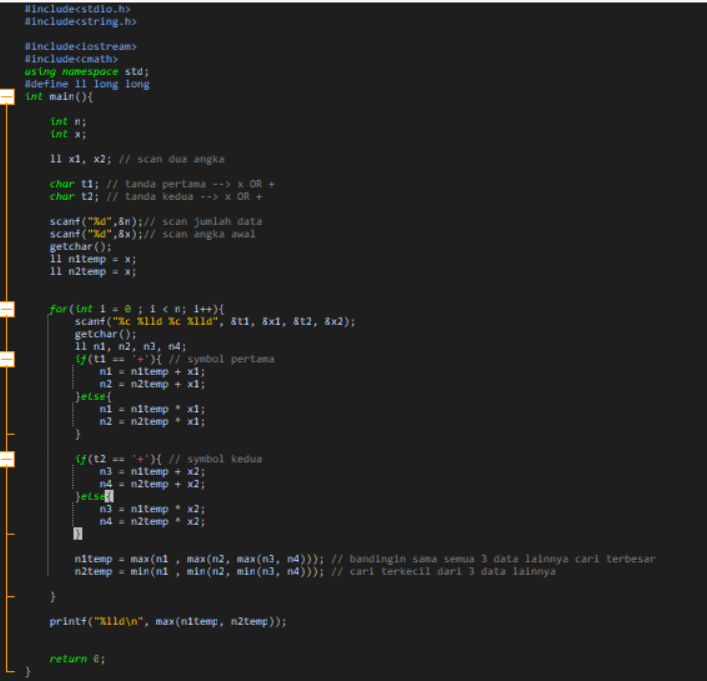
Happiest = 4;

**Maka happiest ( kesenangan terbanyak ) adalah 4**

Alasan tidak tersolving saat INC berlangsung adalah Code yang dikerjakan sudah mendapatkan hasilnya yang sesuai dengan sample inputan di soal namun sistem tetap menetapkan WA, sehingga membingungkan kami untuk men-debug code yang sudah kami testing ( hasil output sudah sesuai) namun tidak mengetahui titik kesalahan berada di area yang mana. Dan faktor keterbatasan waktu juga menjadi faktor lainnya karena banyaknya soal yang ada membingungkan kami untuk men-debug code kami yg WA atau mencoba untuk mengerjakan soal yang lainnya.

**#2 Unsolved Problem G Plus OR Times**





**Penjelasan Algoritma Unsolved Problem G (PLUS OR TIMES)**

1. User memasukkan inputan banyak data (n) dan angka default awal (x).
2. User dapat memasukkan character1 ( ‘+’ OR ‘x’ ) , angka 1 ( can be negative ), character2 ( ‘+’ OR ‘x’ ) , angka 2 ( can be negative ).
3. Loop Segment sesuai banyak data (n), dari angka default awal(x) akan ditambah / dikali dengan angka1 dan angka 2 sesuai dengan tanda 1 dan tanda 2 kedua tanda ini harus dicoba sesuai dengan angkanya dan disimpan dalam sebuah variabel n1, n2, n3, n4.
4. Di setiap akhir iterasi hanya tersisa dua angka yaitu angka terbesar dari n1, n2, n3, n4 dan angka terkecil dari n1, n2, n3, n4 . Angka terkecil disimpan karena ada kemungkinan akan dikalikan dengan bilangan negative sehingga memungkinkan menjadi kandidat angka terbesar. Ulangi Langkah 2 , 3 , 4.
5. Hasil output dari pengecekan hasil maksimal angka terkecil dan angka terbesar.

**Sample Input #1**

3 123

+ 100 x 2

+ -100 x -2

+ 0 + 0

**Iterasi 1 (123)**

N1 = 123 + 100 = 223

N2 = 123 x 2 = 246

N3 = 123 x 2 = 246

N4 = 123 x 2 = 246

Max = 246

Min = 223

**Iterasi 2 (223 246)**

N1 = 223 - 100 = 123

N2 = 246 - 100 = 146

N3 = 223 x -2 = -446

N4 = 246 x -2 = -492

Max = 146

Min = -492

**Iterasi 3 (146 -492)**

N1 = 146 + 0 = 146

N2 = -492 + 0 = -492

N3 = 146 + 0 = 146

N4 = --492 + 0 = -492

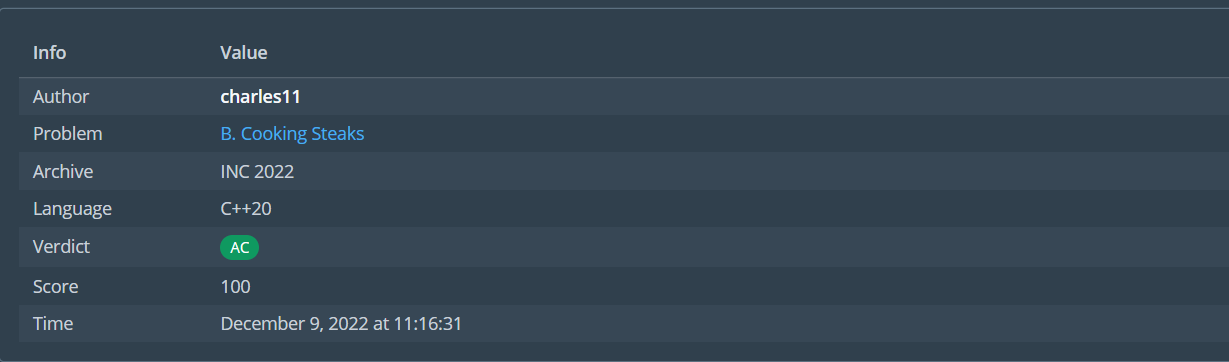
Max = 146

Min = -492

**Print nilai yang lebih besar antara nilai Max dan nilai Min**

Alasan tidak tersolving saat INC berlangsung adalah Code yang dikerjakan sudah mendapatkan hasilnya yang sesuai dengan sample inputan di soal namun sistem tetap menetapkan WA, sehingga membingungkan kami untuk men-debug code yang sudah kami testing ( hasil output sudah sesuai) namun tidak mengetahui titik kesalahan berada di area yang mana. Dan faktor keterbatasan waktu juga menjadi faktor lainnya karena banyaknya soal yang ada membingungkan kami untuk men-debug code kami yg WA atau mencoba untuk mengerjakan soal yang lainnya.

**#3 Unsolved Problem B Cooking Steaks**

****



Penjelasan Algoritma Problem B

1. user input banyaknya data (n), input waktu yang diperlukan steak (i) ke steak(i+1) -> time sebanyak n-1, input stok (stk) yang tersedia sebanyak n dan input orderan(order) sebanyak n masing masing berhubungan dengan stk(i) masing-masing.

2. Loop Segment sebanyak n kali dari paling akhir ke awal, apabila stok(i) dapat memenuhi semua kebutuhan pada order (i) maka lanjut ke orderan selanjutnya. Namun apabila stok [i] tidak dapat memenuhi seluruh orderan [i] , maka kita simpan kekurangannya stok lalu ambil stok dari stok[i-1] lalu dikali dengan waktu yang diperlukan time[i-1] agar dapat menjadi stok[i]. Waktu yang diperlukan ini yang akan menjadi waktu minimal yang ditanyakan.

3. Ulangi langkah ke dua sampai mencapai stok pertama apabila stok pertama masih dapat memenuhi orderan maka output waktu minimal, namun apabila stok pertama sudah tidak dapat memenuhi orderan pada orderan pertama maka output -1

**Sample Input #1**

**Variabel time[n-1] , stok[n], order[n]**

3

1 2

2 2 3

0 1 5

**Iterasi 1**

**Stok[2] = 3**

**Order[2] = 5**

**// ambil kekurangan dari stok[1]**

KurangStok = 5 – 3 = 2

Waktu = 2 \* 2 = 4;

Stok[1] = 2 - KurangStok = 0

**Iterasi 2**

**Stok[1] = 0**

**Order[1] = 1**

**// ambil kekurangan dari stok[0]**

KurangStok = 1 – 0 = 1

Waktu = 4 + (1 \* 1) = 5;

Stok[0] = 2 - 1 = 1

**Iterasi 3**

**Stok[0] = 1**

**Order[0] = 0**

**Waktu = 5**

**// karena stok pertama masih melebihi orderan pertama**

Return waktu; // 5 seconds