BEEFEST 2019

Bina Nusantara Programming Contest For High School Students

25 August 2019

PROBLEM SOLUTIONS

BNPC The 2019 Bina Nusantara Programming Contest for High School Students

hosted by

BINUS UNIVERSITY

Problem Authors

Special Thanks to problem author :

PROBLEM		AUTHOR
A	Bosu Again	Andreas Cendranata
B	3 Plus 1	William Gunawan Eka
C	Zero Two	William Gunawan Eka
D	Elemen Baru	Rafael Herman Yosef
E	Rbit	William Gunawan Eka
F	Dokemon Orange	Andreas Cendranata
G	Sesi Belajar	William Gunawan Eka
H	Tugas Kelompok	Clarissa Arifin
I	Mencari Tempat Tinggal	Jasson Liudy
J	Makhluk Kaki Banyak	Rafael Herman Yosef

A – Bosu Again

Soal yang sama dengan soal Bosu di kualifikasi, hanya berbeda constraint

Ringkasan: Mencari nilai dari 1 + 2 + 3 + ... + N

Solusi: menggunakan rumus jumlah dari 1-N = ((1+N) * N) / 2

- Dapat menggunakan tipe data long long maupun unsigned long long

Solusi dengan perulangan akan mendapatkan TIMELIMIT

Tag: Ad-hoc
Kompleksitas: O(1) untuk setiap kasus

B – 3 Plus 1

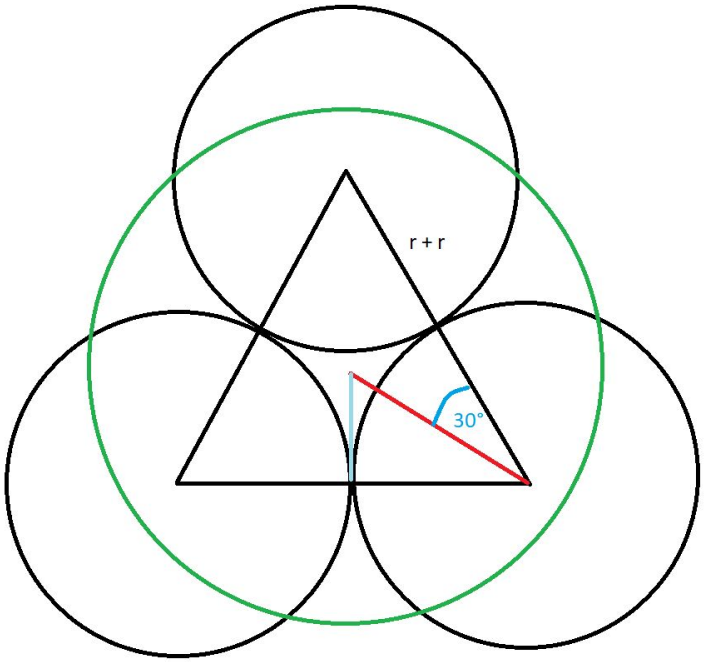
Ringkasan: Diberikan 3 bola identik dan 1 bola lainnya yang disusun sesuai permintaan. Tentukan tinggi dari bangun tersebut

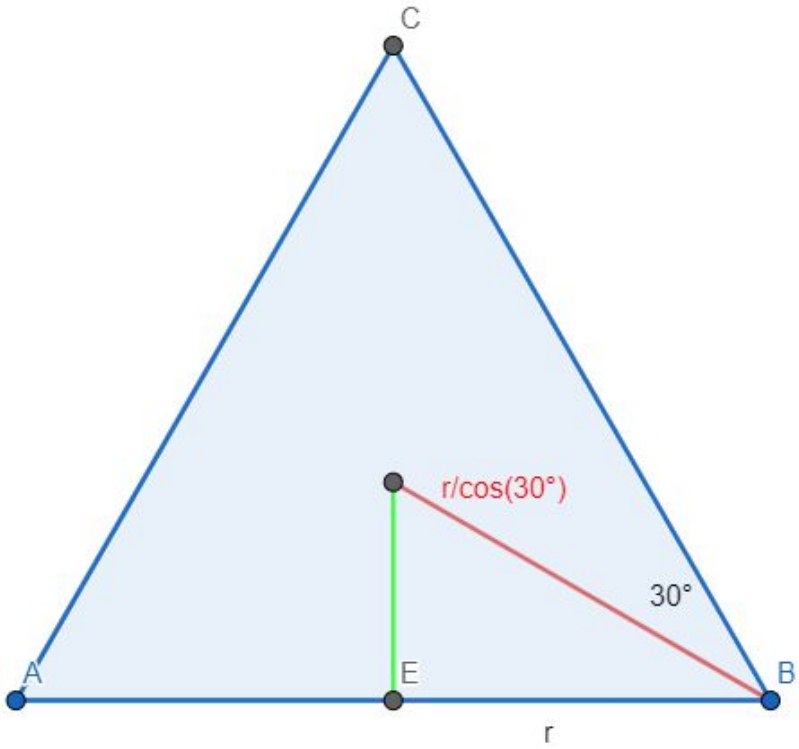
Observasi 1: Jarak antar pusat bola yang bersebelahan adalah jumlah dari kedua jari-jari bola tersebut

Observasi 2: Pusat bola ke-4 terletak tepat di tengah susunan 3 bola bawah

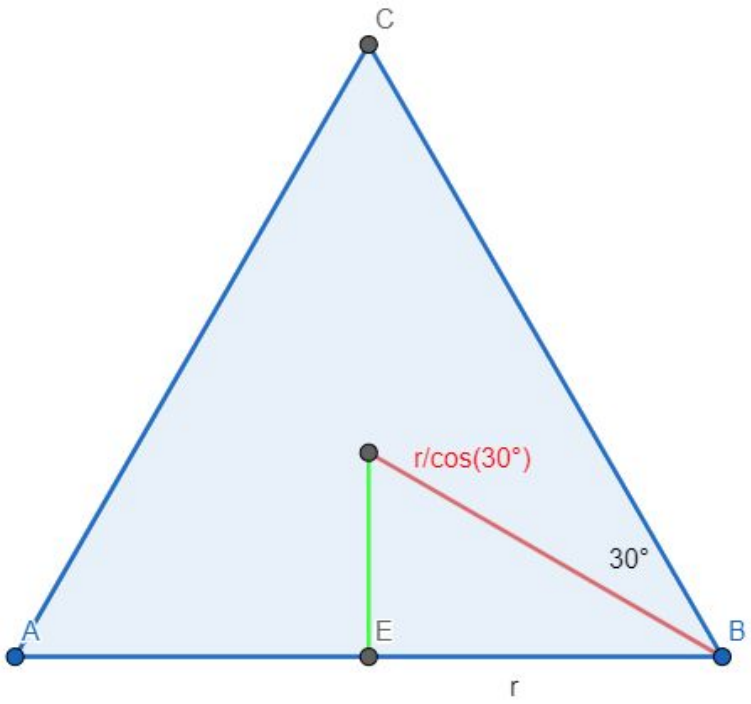
Observasi 3: Susunan 3 bola bawah membentuk segitiga sama sisi dengan sudut 60°

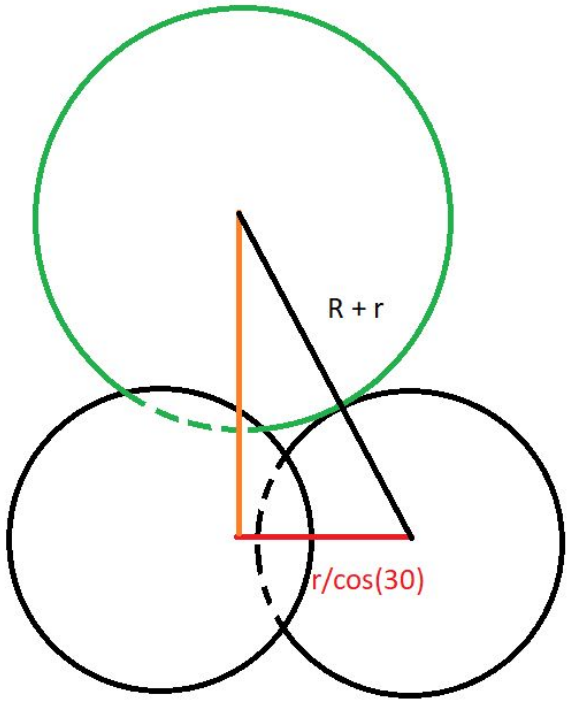
B – 3 Plus 1





B – 3 Plus 1





B – 3 Plus 1

Tag: Math, Geometry

Kompleksitas: O(1) untuk setiap kasus

C – Zero Two

Ringkasan: Tentukan minimum penukaran dari string biner sehingga terdapat tepat 1 substring “00”

Solusi: menggunakan Dynamic Programming dengan DP[n][m][v][p] di mana:

n: posisi index string ($1 \le n \le 2000$)
m: sisa angka “0” yang harus dimasukkan ($1 \le m \le 2000$)
v: apakah sudah terbentuk “00” ($0 \le v \le 1$)
p: apakah angka sebelumnya merupakan “0” ($0 \le p \le 1$)

C – Zero Two

Transisi: untuk setiap posisi n, kita coba masukkan angka 0 dan angka 1. Cost dari DP bertambah 1 bila angka yang kita masukkan tidak sesuai dengan string awal. Untuk menghindari double counting, kita bisa menghitung salah satu ketidakcocokan saja (entah 0 dengan 1, atau 1 dengan 0).

```
DP[i][j][k][0] = min(DP[i-1][j][k][0],
DP[i-1][j][k][1])

DP[i][j][0][1] = (str[i-1]=='1'?1:0) +
DP[i-1][j-1][0][0]

DP[i][j][1][1] = (str[i-1]=='1'?1:0) +
min(DP[i-1][j-1][0][1], DP[i-1][j-1][1][0])
```

C – Zero Two

Tag: Dynamic Programming

Kompleksitas: $O(N^2)$ untuk setiap kasus

D – Elemen Baru

Ringkasan:

Diberikan N titik yang memiliki arah geraknya masing-masing. Apabila sepasang titik berjarak tidak lebih dari R, maka akan menghasilkan 1 energi. Tentukan energi maksimal yang dapat dihasilkan pada suatu waktu.

Karena jumlah titik maksimal 1000, maka kita dapat mencoba untuk setiap pasang titik yang mungkin, pada rentang waktu kapan akan berjarak tidak lebih dari R.

D – Elemen Baru

Posisi titik i dapat direpresentasikan sebagai fungsi $X_i(t)$ dan $Y_i(t)$ dimana:

$$X_i(t) = x_i + t * dx_i,\ Y_i(t) = y_i + t * dy_i$$

Dengan x_i, y_i adalah posisi awal titik i, dx_i, dy_i, adalah pergerakan titik i, t adalah waktu.

Setiap pasang titik akan menghasilkan energi selama jarak kedua titik di bawah R

$$D(i, j) \leq R$$

Dengan D(i, j) adalah jarak euclidean kedua titik.

$$D(i, j) = \sqrt{(\text{selisih } x)^2 + (\text{selisih } y)^2} \leq R$$

D – Elemen Baru

Kuadratkan kedua ruas

$(\text{selisih } x)^2 + (\text{selisih } y)^2 \leq R^2$

$(x_i + t * dx_i - x_j - t * dx_j)^2 + (y_i + t * dy_i - y_j - t * dy_j)^2 \leq R^2$

$((x_i - x_j) + t * (dx_i - dx_j))^2 + ((y_i - y_j) + t * (dy_i - dy_j))^2 \leq R^2$

$t^2 * (dx_i - dx_j + dy_i - dy_j) + t * \{2 * (dx_i - dx_j) * (x_i - x_j) + 2 * (dx_i - dx_j) * (x_i - x_j)\} + (x_i - x_j)^2 + (y_i - y_j)^2 - R^2 \leq 0$

Bentuk di atas adalah pertidaksamaan kuadrat berbentuk $At^2 + Bt + C \leq 0$

D – Elemen Baru

Kita dapat menentukan rentang t dengan menggunakan quadratic formula (formula A, B, C)

$$t_{1,2} = (-B \pm \sqrt{B^2 - 4AC}) / 2A$$

Anggap $t_1 \leq t_2$, maka rentang yang menghasilkan energi adalah $t_1 \leq t \leq t_2$

Sehingga permasalahan sekarang adalah diberikan beberapa rentang, cari sebuah waktu yang berada dalam rentang terbanyak. Ini adalah permasalahan *maximum interval overlap*.

D – Elemen Baru

Tricky Case:

Ada beberapa pasang titik yang harus diberikan penanganan khusus:

1. Kedua titik memiliki dx, dy yang sama sehingga menyebabkan nilai A = 0. Apabila kedua titik berjarak tidak lebih dari R, maka tambahkan secara manual ke jawaban. Jika lebih dari R, maka abaikan
2. Waktunya berada pada rentang negatif seluruhnya, $t_1 \le t_2 < 0$. Abaikan rentang ini
3. Waktunya berada pada rentang negatif dan positif. $t_1 < 0 < t_2$. Masukkan $0 \le t \le t_2$
4. Tidak ada solusi dari t, karena nilai $B^2 - 4AC < 0$. Abaikan pasangan ini

D – Elemen Baru

Tag: Geometry, Math, Data Structure / Two Pointer

Kompleksitas: $O(N^2 \log N)$

E – Rbit

Ringkasan: Cari N serta urutan penjumlahan dan pengurangan agar hasil deret tepat P

Observasi 1: Untuk jarak lompatan awal N dan semua lompatan berarah ke kanan, total jarak yang ditempuh adalah N*(N+1)/2

Observasi 2: Rbit tidak mungkin sampai ke-P bila N*(N+1)/2 < P

Observasi 3: Untuk suatu urutan lompatan dengan jarak tempuh X, bila terdapat lompatan ke kanan dengan jarak L, maka kita bisa mengubah lompatan tersebut ke kiri sehingga jarak tempuh menjadi X - 2*L

E – Rbit

Solusi: Kita bisa melakukan brute force untuk mencoba setiap N. Jika selisih N*(N+1)/2 dengan P genap, Rbit pasti bisa sampai ke P. Sehingga untuk N tersebut, kita bisa secara greedy mengubah satu per satu lompatan ke kiri hingga jarak tempuh menjadi P.

Tag: Brute Force + Greedy
Kompleksitas: O(P) untuk setiap kasus

F – Dokemon Orange

Ringkasan: Cari nilai maksimal dari fungsi f(a,b) untuk $L \le a \le b \le R$

Observasi 1: Fungsi yang diberikan dapat disederhanakan menjadi

$$f(a,b) = a \text{ XOR } (a+1) \text{ XOR } \dots \text{ XOR } (b-1) \text{ XOR } b$$

Observasi 2: Hasil dari

$$4k \text{ XOR } (4k+1) \text{ XOR } (4k+2) \text{ XOR } (4k+3) = 0$$

Solusi: Berdasarkan 2 observasi di atas, kita dapat mencoba 4 kemungkinan a (posisi awal) dan 4 kemungkinan b (posisi akhir).

Tag: Ad-hoc

Kompleksitas: O(4*4)

G – Sesi Belajar

Ringkasan: Untuk setiap orang, jika orang disebelah kirinya belajar selama x, maka orang tersebut bisa ikutan belajar maksimal x.

Solusi: Kita bisa melakukan greedy, untuk setiap orang, jika orang di sebelah kirinya lebih lama belajar, maka tidak menambah sesi belajar. Jika orang di sebelah kirinya lebih sedikit belajar, maka kita perlu menambah sesi belajar sebanyak selisih dari orang sekarang dan orang sebelumnya

Tag : Greedy
Kompleksitas : O(n)

H – Tugas Kelompok

Ringkasan: Tentukan waktu minimum untuk menyelesaikan N tugas kelompok dengan M anggota kelompok. 1 anggota dapat memilih 2 titik L dan R dan mengerjakan soal dari L sampai R. Suatu soal dapat dikerjakan lebih dari 1 anggota.

Observasi 1: Suatu soal yang dikerjakan lebih dari 1 anggota hanya menambah waktu yang diperlukan untuk menyelesaikan tugas kelompok tersebut. Maka setiap soal sebaiknya dikerjakan 1 anggota saja.

Observasi 2: Apabila dalam waktu X suatu tugas kelompok dapat diselesaikan, maka tugas kelompok tersebut dapat diselesaikan dalam waktu lebih dari X.

H – Tugas Kelompok

Solusi: Binary Search untuk mencari jawaban.

Selama anggota ke-i masih dapat mengerjakan sebuah soal dengan waktu yang kurang dari sama dengan jawaban, maka soal dapat dikerjakan anggota tersebut. Selain itu, anggota ke-i+1 dapat mengerjakan soal tersebut. Bila anggota yang dibutuhkan lebih kecil sama dengan M, maka jawaban tersebut merupakan salah satu kandidat jawaban.

Tag : Binary Search
Kompleksitas : O(n log n)

I – Mencari Tempat Tinggal

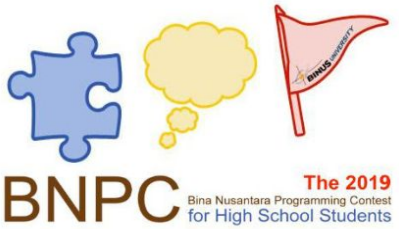
Ringkasan: Mencari persimpangan yang meminimalisasi jarak terjauh ke setiap tempat tujuan.

Step 1: Setiap leaf yang bukan tempat tujuan dibuang karena leaf tersebut tidak akan dikunjungi. Lakukan secara berulang hingga semua leaf adalah tempat tujuan. Bisa dilakukan dengan *Breadth-First Search* (BFS).

Step 2: Lakukan BFS dengan *priority queue* dari setiap leaf, setiap kali mengevaluasi sebuah jalan kita mengurangi jalan yang terhubung dengan persimpangan tersebut. *Priority queue* hanya boleh berisi leaf.

Step 3: persimpangan terakhir yang berada dalam *priority queue* adalah tempat tinggal paling optimal.

I – Mencari Tempat Tinggal



Tag: Graph Traversal (BFS)

Kompleksitas: O(N * log(N) + M)

J - Makhluk Kaki Banyak

Ringkasan: Terdapat makhluk berkaki N yang ingin memakai celana, kaos kaki, dan sepatu. Sepatu hanya bisa dipakai pada suatu kaki apabila pada kaki tersebut telah dipakai celana dan kaos kaki. Tentukan banyak cara pemakaiannya.

J - Makhluk Kaki Banyak

Untuk sekarang, anggap saja pemasangan celana, kaos kaki, dan sepatu pada kaki ke i disimbolkan dengan x_i, x_i, x_i. (x_i nya ada 3 dan sama semua)

Apabila x_i baru muncul sekali, anggap sebagai pemasangan celana, jika x_i muncul untuk kedua kalinya, anggap sebagai pemasangan kaos kaki, dan x_i terakhir sebagai sepatu.

Selain itu dapat dibaca sebagai x_i pertama, anggap sebagai kaos kaki dan, x_i kedua untuk celana.

Sehingga untuk N karakter terdapat 2^N cara pembacaan

J - Makhluk Kaki Banyak

Permasalahan soal tersebut dapat diubah menjadi, “Diberikan sebuah kata yang tersusun dari huruf $x_1x_1x_1x_2x_2x_{2...}x_Nx_Nx_N$ tentukan banyaknya kata berbeda yang dapat dibentuk.”

Solusi dari permasalahan ini adalah dengan menggunakan permutasi dari 3N yang terdiri dari N karakter dimana setiap karakter muncul 3 kali.

Secara rumus: $3N! / (3!)^N$

Karena ada 2^N cara pembacaan, maka kalikan jawaban tersebut dengan 2^N. Sehingga jawabannya: $3N! * 2^N / (3!)^N = 3N! / 3^N$

J - Makhluk Kaki Banyak

Bagaimana cara menghitung 3N! / 3^N ? Ini pseudocodenya

```
result = 1
for i = 1 to 3N
   if (i mod 3 = 0)
      result = (result * (i / 3)) modulo MOD
   else
      result = (result * i) modulo MOD
```

Coret secara manual :)

Tag: Mathematics

Kompleksitas: O(3N) untuk setiap testcase

Questions?