A. Teks Fibonacci

Nilai k yang mencapai 2^{31}-1 membuat kita tidak mungkin menggenerate string teks fibonacci karena akan membutuhkan space memori yang besar. Alternatifnya, kita bisa menyelesaikan soal ini dengan menggunakan struktur rekursi.

Pertama kita cari dahulu nilai k ini jatuh pada kata fibonacci yang ke berapa. Panjang setiap kata fibonacci dapat dihitung sesuai deret fibonacci ($|S_0| = F_1 = 1$, $|S_1| = F_2 = 1$, $|S_2| = F_3 = 2$, $|S_3| = F_4 = 3$, dst). Setelah itu, karena kata fibonacci dibentuk oleh rekursi, maka mencari huruf yang membentuk kata tersebut juga bisa dilakukan secara rekursi.

B. BeSaR DaN KeCiL

Ini adalah soal termudah dalam kontes. Anda hanya diminta untuk mengubah huruf kecil menjadi kapital. Pada C/C++, karena indeks dimulai dari 0, maka jika indeks genap ubah ke huruf kapital. Pada Pascal, karena indeks dimulai dari 1, maka jika indeks ganjil ubah ke huruf kapital.

C. Rentang Terbesar

Tidak ada gunanya mencari subset, karena akan lebih baik jika semua bilangan dipilih. Masalahnya tinggal seberapa jauh kita bisa memperbesar rentang w yang bisa dibentuk dengan angka yang tersedia.

Pertama, input harus kita sort terlebih dahulu. Kemudian angka 1 harus ada di dalam himpunan (agar bisa membentuk nilai 1). Misalnya saat ini kita sedang memeriksa bilangan ke i (anggap arr[i]), dan nilai w sementara adalah x. Maka w bisa kita perlebar menjadi x + arr[i] dengan syarat semua angka dari 1..arr[i]-1 bisa dibentuk, dan ini ekuivalen dengan mengecek apakah x >= arr[i]-1. Mengapa? Karena ini menandakan bahwa kita sudah bisa membentuk minimal hingga arr[i]-1, dan dengan menambahkan ar[i], maka w menjadi arr[i]+x. Dijamin bahwa semua angka dari arr[i] hingga arr[i]+x bisa dibentuk (dengan cara menambahkan ar[i] dengan subset yang membentuk nilai sisanya memakai bilangan-bilangan sebelum arr[i]). Namun bagaimana jika syarat tadi tidak terpenuhi? maka kita hentikan pengecekannya, karena tidak mungkin w bisa diperbesar lagi.

D. Merakit Komputer

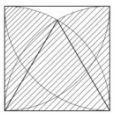
Soal ini terlihat mudah untuk dipahami, namun akan membutuhkan implementasi yang cukup panjang. Jumlah toko maksimal 10, dengan demikian kita bisa melakukan *bruteforce* pada toko-toko yang ingin digunakan jasanya (2^{10}). Jika toko-toko yang mau digunakan jasanya sudah ditentukan, maka pembelian barang bisa dilakukan secara *greedy* dengan mengambil komponen termurah untuk merakit komputer.

E. Mario Bros

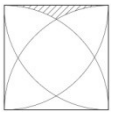
Soal ini bisa diselesaikan dengan sedikit modifikasi pada pencarian BFS (Breath First Search). Flag `visited` juga kita gunakan untuk dinding, agar ketika dinding tersebut dihancurkan (sehingga bisa dilewati), kita bisa menghidupkan kembali state di dinding tersebut. Jadi ketika kita meledakan suatu dinding, kita tinggal memeriksa apakah dinding tersebut sebelumnya pernah "dikunjungi", jika ya, maka hidupkan kembali state di dinding tersebut.

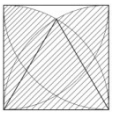
F. Kaca Patri

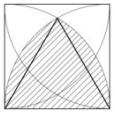
Terlebih dahulu kita perlu mencari cara menghitung luas daerah berwarna (berarsir).



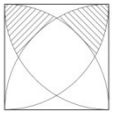
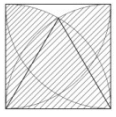
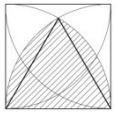
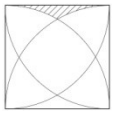
= luas segitiga + luas dua juring
= ½ * s * ½ s √3 + 2 * 30/360 * π * s * s
= ¼ * s * s * √3 + 1/6 * π * s * s



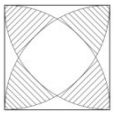
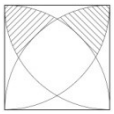
= luas persegi - 
= s * s - (¼ * s * s * √3 + 1/6 * π * s * s)



= luas dua juring – luas segitiga
= 2 * 60/360 * π * s * s – ½ * s * ½ s √3
= 1/3 * π * s * s – ¼ * s * s * √3

 =  -  - 2 * 

= ¼ * s * s * √3 + 1/6 * π * s * s – (1/3 * π * s * s – ¼ * s * s * √3) – 2 * (s * s - (¼ * s * s * √3 + 1/6 * π * s * s))
= s * s * √3 + 1/6 * π * s * s – 2 * s * s
= s * s (1/6 * π + √3 – 2)

 = 2 * 

= 2 * s * s (1/6 * π + √3 – 2)

= s * s * (1/3 * π + 2√3 – 4)

Setelah kita dapatkan rumus mencari luas daerah berwarna dari s, maka yang perlu kita lakukan berikutnya adalah *binary search* pada bilangan bulat s dan cari s terbesar yang menghasilkan luas tidak lebih dari L.

G. Jumlah Pembagi

Pertama, N kita faktorkan terlebih dahulu sehingga berbentuk $N = p_1^{a1} * p_2^{a2} * p_3^{a3} * \ldots * p_k^{ak}$ (dimana p_i adalah faktor prima dari N). Kemudian untuk mencari jumlah pembagi (sum of divisor), kita bisa menggunakan persamaan:

$SOD = (p_1^0 + p_1^1 + p_1^2 + \ldots + p_1^{a1}) * (p_2^0 + p_2^1 + p_2^2 + \ldots + p_2^{a2}) * \ldots * (p_k^0 + p_k^1 + p_k^2 + \ldots + p_k^{ak})$

Masalah berikutnya adalah bagaimana cara menghitung $\sum_{j=0}^{ai} Pi^j$ dengan cepat, mengingat pangkat a_i bisa mencapai ratusan juta. Ada (setidaknya) dua cara untuk menyelesaikan masalah ini, yang pertama adalah dengan *divide and conquer*.

```
f(a,n) {  // menghitung nilai dari a^0 + a^1 + ... a^n
    if (n == 0) return 1;
    if ( n mod 2 == 1)  {
        x = f(a,n-1);
        return x + x * (a - 1) + 1 ;
    }
    else {
        x = f(a,n/2);
        return x * (x * (a - 1) + 1) + x;
    }
}
```

Cara kedua adalah dengan menggunakan rekursi yang dimodelkan sebagai matrix $A = (1 \;\; 1) \begin{pmatrix} 1 & 0 \\ a & a \end{pmatrix}^n$, di mana elemen (1,1) dari matrix A adalah f(a,n) dan elemen (1,2) adalah a^n. Di sini teknik *divide and conquer* kembali digunakan untuk memangkatkan matrix tersebut.

Hati-hati terhadap kasus 0^n yang outputnya adalah 0.

H. Drum Minyak Pak Ricat

Soal ini adalah tipikal soal *dynamic programming*, dimana kita diharuskan mencari solusi optimal dari berbagai solusi yang ada. Soal ini dapat kita selesaikan dengan menjalankan dua kali *dynamic programming knapsack* untuk mendapatkan jawabannya.

Kita akan menjalankan *DP knapsack* untuk pertama kali untuk mendapatkan semua kemungkinan isi dari drum dengan harga yang termurah, dengan relasi rekursi sbb :

C_d = kapasitas maksimal drum
M = jumlah paket
C_i = kapasitas pengisian paket ke i
P_i = harga paket ke i

$$f_1(C_d) = \min_{i=1..M} f_1(C_d - C_i) + P_i$$

Kemudian untuk *DP knapsack* kedua, kita akan memakai hasil dari *DP* pertama tadi untuk mendapatkan solusi optimal yang kita inginkan, dengan relasi rekursi sbb :

N = jumlah minyak yang akan di distribusikan
C_d = kapasitas maksimal drum
P_d = harga 1 drum
$f_1(i)$ = harga optimal dari knapsack pertama untuk kapasitas i dalam 1 drum

$$f_2(N) = \min_{i=1..Cd} f_2(N - i) + f_1(i) + P_d$$

I. Jumlah Pangkat

a^b mod 10 bisa dihitung dengan cepat. Misalnya 2^n, digit terakhir dari pemangkatan 2 adalah 1, 2, 4, 8, 6, 2, 4, 6, 8, 2, …, dst. 2, 4, 8 dan 6 adalah empat digit yang akan terus berulang secara periodik. Dengan demikian kita bisa menghitung 2^n dengan memodulo n dengan 4.

J. Ordo Keprimaan

Soal ini bisa diselesaikan dengan menyimulasikan apa yang diminta. Untuk menggenerate bilangan prima bisa menggunakan metode *sieve eratosthenes*.