Instituto Politecnico Nacional

ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”

ANÁLISIS DE ALGORITMOS

DISEÑO DE SOLUCIONES CON PROGRAMACIÓN DINÁMICA

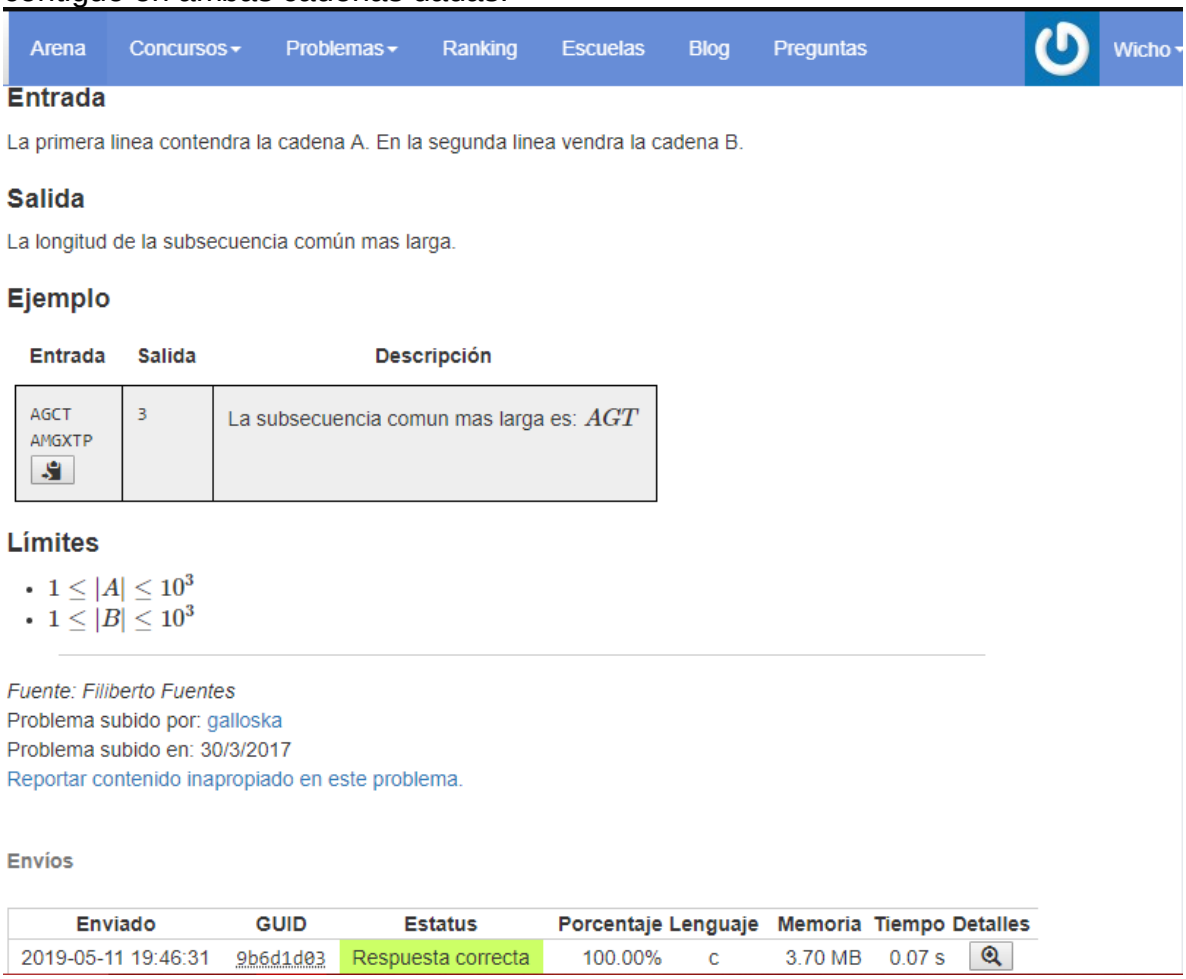
PROFE: Edgardo Adrián Franco Martínez

ALUMMNO: Rojas Alvarado Luis Enrique

GRUPO: 3CM4

I. Longest common subsequence.

Se dan dos cadenas A y B de longitudes n y m, y nuestro trabajo es encontrar la longitud de la sub-secuencia común más larga (LCS). Se define como la secuencia de caracteres más larga que aparece de izquierda a derecha, no necesariamente contiguo en ambas cadenas dadas.



Arena Concursos Problemas Ranking Escuelas Blog Preguntas Wicho

Entrada

La primera linea contendra la cadena A. En la segunda linea vendra la cadena B.

Salida

La longitud de la subsecuencia común mas larga.

Ejemplo

Entrada	Salida	Descripción
AGCT AMGXTP	3	La subsecuencia comun mas larga es: AGT

Límites

- $1 \leq |A| \leq 10^3$
- $1 \leq |B| \leq 10^3$

Fuente: Filiberto Fuentes
Problema subido por: galloska
Problema subido en: 30/3/2017
Reportar contenido inapropiado en este problema.

Envíos

Enviado	GUID	Estatus	Porcentaje	Lenguaje	Memoria	Tiempo	Detalles
2019-05-11 19:46:31	9b6d1d03	Respuesta correcta	100.00%	c	3.70 MB	0.07 s	

Para ésta solución se utilizó la programación dinámica. En el momento de que la longitud del arreglo 1 sea diferente de la longitud del arreglo 2 se toma el máximo de la misma función, solo que con el índice menos uno, de igual manera para el arreglo 2 (Sus respectivos índices).

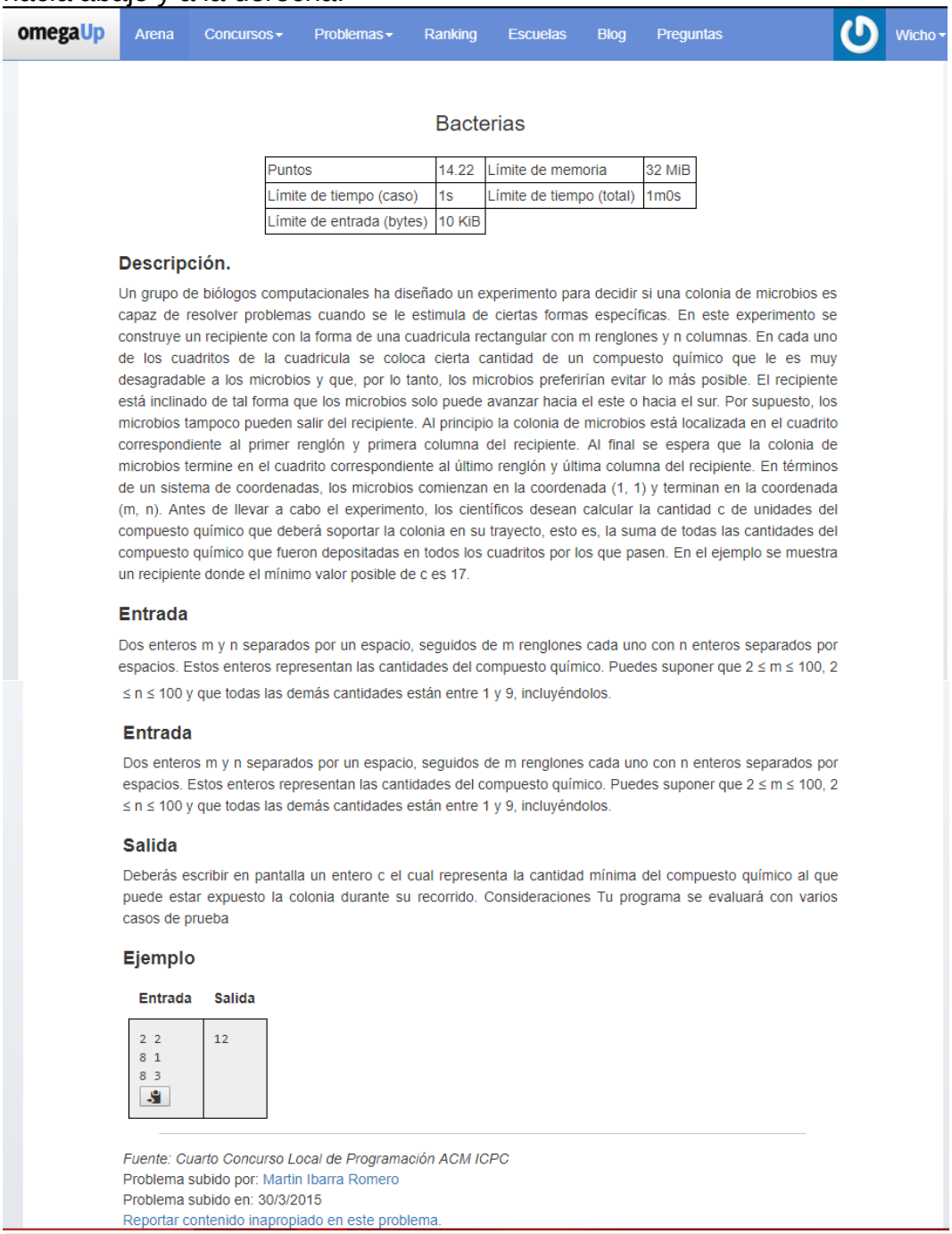
En el caso de que sean iguales el LCS será 1 + la función con los 2 índices -1, y si tenemos los índices en 0 significa que aún no hemos leído la cadena completa, o una de las dos cadenas no ha sido recorrida completamente.

Simplemente es poner en práctica lo que el profesor nos enseñó en clase, ya que éste problema lo resolvimos en grupo siendo la complejidad de éste problema de O(nm)

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int max(int algo1, int algo2){
  if (algo1<algo2) {
    return algo2;
  }
  else{
    return algo1;
  }
}

int lcs(char c1[],char c2[], int i, int j,int tam1, int tam2,int mat[tam1][tam2]){
  if (i==tam1 || j==tam2) {
    return 0;
  }
  if (c1[i]==c2[j]) {
    if (mat[i][j] == -1) {
      mat[i][j]=1+lcs(c1,c2,i+1,j+1,tam1,tam2,mat);
    }
    return mat[i][j];
  }else{
    if (mat[i][j] == -1) {
      mat[i][j]=max(lcs(c1,c2,i,j+1,tam1,tam2,mat),lcs(c1,c2,i+1,j,tam1,tam2,mat));
    }
    return mat[i][j];
  }
}

int main(int argc, char const *argv[]) {
  char c1[1000], c2[1000];
  int i,j;
  scanf("%s",&c1);
  scanf("%s",&c2);
  int tam1 = strlen(c1);
  int tam2 = strlen(c2);
  int mat[tam1][tam2];

  for (i = 0; i < tam1; i++) {
    for (j = 0; j < tam2; j++) {
      mat[i][j]=-1;
    }
  }
  printf("%i",lcs(c1,c2,0,0,tam1,tam2,mat));
  return 0;
}
```

II. Bacterias.

Si tenemos una cuadrícula de m x n, dónde cada cuadro tiene un costo asociado cuando pasa por él. Lo que nos interesa es hallar el costo minimo de la esquina superior izquierda a la esquina inferior dedrecha y desde ahí podemos movernos hacia abajo y a la derecha.



omegaUp Arena Concursos Problemas Ranking Escuelas Blog Preguntas Wicho

Bacterias

Puntos	14.22	Límite de memoria	32 MiB
Límite de tiempo (caso)	1s	Límite de tiempo (total)	1m0s
Límite de entrada (bytes)	10 KiB		

Descripción.

Un grupo de biólogos computacionales ha diseñado un experimento para decidir si una colonia de microbios es capaz de resolver problemas cuando se le estimula de ciertas formas específicas. En este experimento se construye un recipiente con la forma de una cuadricula rectangular con m renglones y n columnas. En cada uno de los cuadritos de la cuadricula se coloca cierta cantidad de un compuesto químico que le es muy desagradable a los microbios y que, por lo tanto, los microbios preferirían evitar lo más posible. El recipiente está inclinado de tal forma que los microbios solo puede avanzar hacia el este o hacia el sur. Por supuesto, los microbios tampoco pueden salir del recipiente. Al principio la colonia de microbios está localizada en el cuadrito correspondiente al primer renglón y primera columna del recipiente. Al final se espera que la colonia de microbios termine en el cuadrito correspondiente al último renglón y última columna del recipiente. En términos de un sistema de coordenadas, los microbios comienzan en la coordenada (1, 1) y terminan en la coordenada (m, n). Antes de llevar a cabo el experimento, los científicos desean calcular la cantidad c de unidades del compuesto químico que deberá soportar la colonia en su trayecto, esto es, la suma de todas las cantidades del compuesto químico que fueron depositadas en todos los cuadritos por los que pasen. En el ejemplo se muestra un recipiente donde el mínimo valor posible de c es 17.

Entrada

Dos enteros m y n separados por un espacio, seguidos de m renglones cada uno con n enteros separados por espacios. Estos enteros representan las cantidades del compuesto químico. Puedes suponer que 2 ≤ m ≤ 100, 2 ≤ n ≤ 100 y que todas las demás cantidades están entre 1 y 9, incluyéndolos.

Entrada

Dos enteros m y n separados por un espacio, seguidos de m renglones cada uno con n enteros separados por espacios. Estos enteros representan las cantidades del compuesto químico. Puedes suponer que 2 ≤ m ≤ 100, 2 ≤ n ≤ 100 y que todas las demás cantidades están entre 1 y 9, incluyéndolos.

Salida

Deberás escribir en pantalla un entero c el cual representa la cantidad mínima del compuesto químico al que puede estar expuesto la colonia durante su recorrido. Consideraciones Tu programa se evaluará con varios casos de prueba

Ejemplo

Entrada	Salida
2 2 8 1 8 3	12

Fuente: Cuarto Concurso Local de Programación ACM ICPC
Problema subido por: Martin Ibarra Romero
Problema subido en: 30/3/2015
Reportar contenido inapropiado en este problema.

Enviado	GUID	Estatus	Porcentaje	Lenguaje	Memoria	Tiempo	Detalles
2019-05-14 19:17:04	7ef81e15	Respuesta correcta	100.00%	cpp11	3.23 MB	0.02 s	

Teniendo varias formas de movernos ya sea llegando por la izquierda o llegando por arriba. Por lo tanto el costo para llegar a la posición que queramos es el minimo de los costos restando cada índice. Simplemente es la suma de los valores que están a la izquierda o arriba de la casilla.

```c
#include <stdio.h>

int min(int a, int b) { return (a < b)? b : a; }

int main (){
    int i, j, m, n;

    scanf("%d %d",&m, &n);

    int cuad[m][n], bact[m][n];

    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            scanf("%d",&cuad[i][j]);
            bact[i][j]=0;
        }
    }

    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            if(i==0 && j==0){
                bact[i][j]=cuad[i][j];
            }
            else if(j==0){
                bact[i][j]=bact[i-1][j]+bact[i][j];
            }
            else if(i==0){
                bact[i][j]=bact[i][j-1]+bact[i][j];
            }
            else{
                bact[i][j]=min(bact[i][j-1],bact[i-1][j]+cuad[i][j]);
            }
        }
    }
    printf("%d \n",bact[m-1][n-1]+1);
    return 0;
}
```

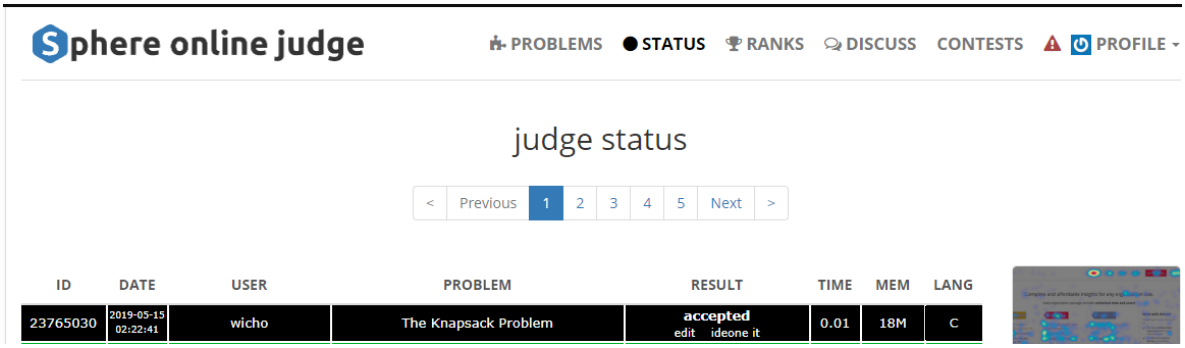
III. The knapsack problem.

Tenemos una mochila con capacidad S y tenemos n elementos que deseamos guardar en la mochila. Ca elemento tiene un peso w y un valor v. Queremos un

subconjunto de esos elementos en la mochila tal que el valor total quede maximizado no sobrepasando su capacidad y no podemos partir los objetos: Se toman o no se toman.

Modelo de implementación Bottom up.

```c
#include<stdio.h>

int max(int a, int b) { return (a > b)? a : b; }

int knapSack(int W, int wt[], int val[], int n)
{
   int i, w;
   int K[n+1][W+1];

   for (i = 0; i <= n; i++)
   {
       for (w = 0; w <= W; w++)
       {
           if (i==0 || w==0)
               K[i][w] = 0;
           else if (wt[i-1] <= w)
                 K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
           else
                 K[i][w] = K[i-1][w];
       }
   }

   return K[n][W];
}


int main()
{
    int i, s, n;
    scanf("%d %d",&s,&n);
    int wt[n], val[n];
    for(i=0;i<n;i++){
        scanf("%d %d",&wt,&val);
    }
    printf("%d", knapSack(n, wt, val,n) );
    return 0;
}
```



La solución se basa en tomar todos los subconjuntos de elementos posibles y er si caben en la mochila. Teniendo el valor máximo de la mochila podemos considerar los primeros objetos con capacidad dentro del rango de la capacidad disponible en la mochila, si los objetos son 0 es cuando no tomamos nada.

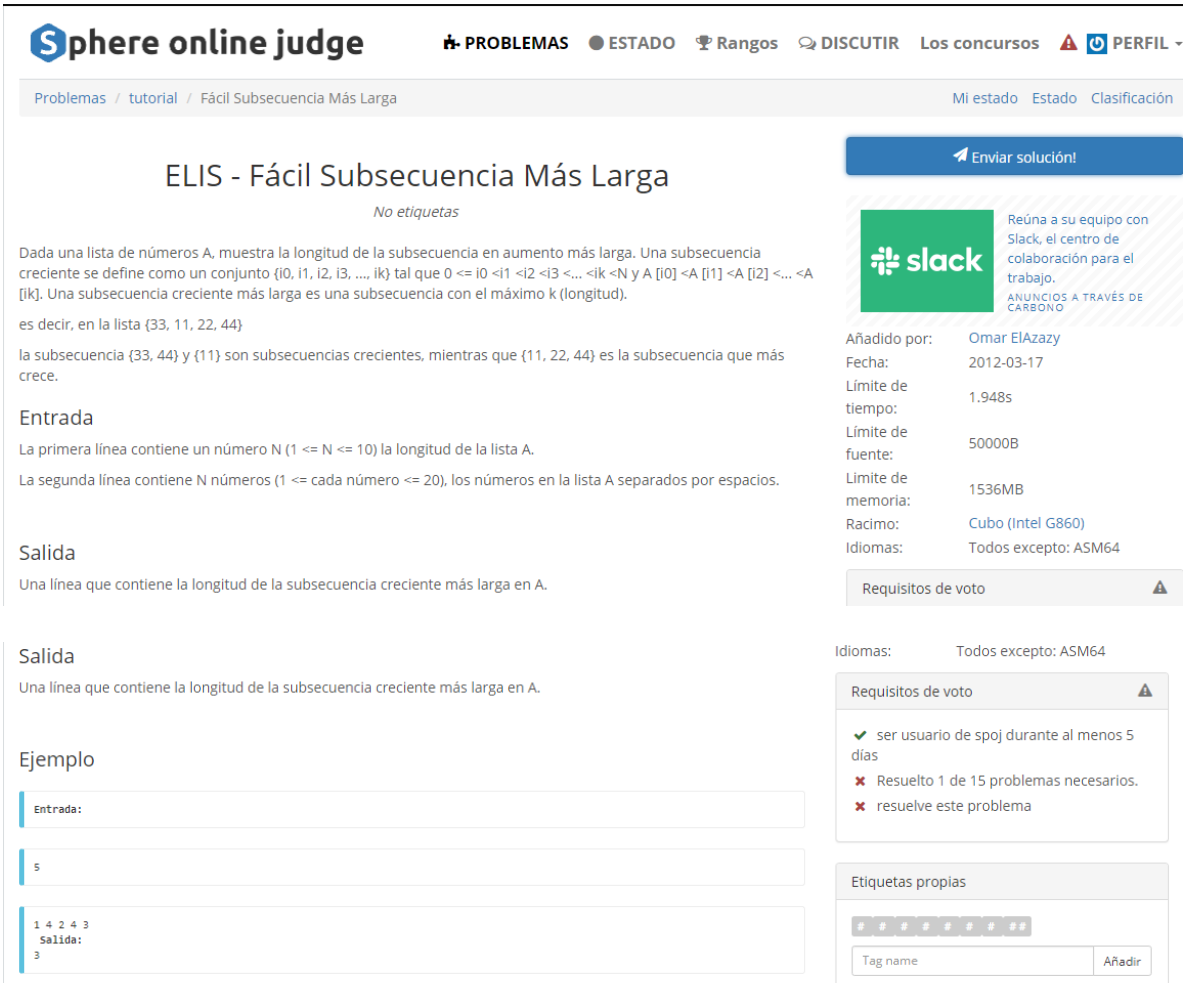
Por otra parte podemos tomar el peso del objeto si es menor al de la mochila, pues si es mayor excederemos el peso de la mochila. Si tomamos el objeto, tenemos que restar un objeto del arreglo, restar el peso del objeto al de la mochila y sumar su

valor, si no lo tomamos el valor es igual a el siguiente objeto (se quita el mismo objeto) y el peso de la mochila. Entonces se toma el máximo de esos dos valores.

Siendo el modelo de solución Bottom Up, la complejidad llega a ser O(sn) Porque hacemos una tabla con los valores.

IV. Easy Longest Increasing Sunsequence.

El problema nos dice que tenemos que darle como entrada una lista de números y muestra la longitud de la sub-secuencia en aumento más larga, como entradas se acepta la longitud de la lista y los números de la lista separados por espacios.



Sphere online judge

PROBLEMS STATUS RANKS DISCUSS CONTESTS PROFILE

judge status

< Previous 1 2 3 4 5 Next >

ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
23765044	2019-05-15 02:30:10	wicho	Easy Longest Increasing Subsequence	accepted edit ideone it	0.00	16M	CPP

Para llegar a la solución se declaran dos auxiliares para poder guardar y para las comparaciones más adelante, simplemente se recorre la lista y si se cumple la condición de que si un numero en la posición de la lista es menor al siguiente número se saca el máximo de los 2 y se guarda en los auxiliares, ya que se termina la comparación se agrega a la tabla, siendo ésta diseño de solución Bottom Up, checando cual es máximo para poder retornar un resultado. Que será el que se imprime en pantalla. La complejidad es de $O(n^2)$.

```cpp
#include<iostream>
#include<vector>
using namespace std;

int LIS(vector<int> &lista, vector<int> &num){
    int lis=-1;
    for(int i=0; i<lista.size(); i++){
        int aux=1;
        int auxM=1;
        for(int j=0; j<i; j++){
            if(lista[j]<lista[i]){
                aux=max(1, num[j] + 1);
                auxM=max(aux, auxM);
            }
        }
        num[i]=auxM;
    }
    for(int s=0; s<num.size(); s++)
        lis=max(lis, num[s]);
    return lis;
}

int main(){
    int n;
    cin >> n;
    vector<int> lista(n);
    vector<int> num(n);
    for(int i = 0; i < n; i++)
        cin >> lista[i];

    cout << LIS(lista, num) << "\n";
    return 0;
}
```

V. Mixtures.

Sphere online judge — PROBLEMAS · ESTADO · Rangos · DISCUTIR · Los concursos · PERFIL

Problemas / clásico / Mezclas — Mi estado · Estado · Clasificación

MEZCLAS - Mezclas

#programación dinámica

Harry Potter tiene n mezclas delante de él, dispuestas en una fila. Cada mezcla tiene uno de 100 colores diferentes (los colores tienen números del 0 al 99).

Quiere mezclar todas estas mezclas. En cada paso, tomará dos mezclas que están juntas una junto a la otra, las mezclará y colocará la mezcla resultante en su lugar.

Al mezclar dos mezclas de colores a y b, la mezcla resultante tendrá el color (a + b) mod 100.

Además, habrá algo de humo en el proceso. La cantidad de humo generado al mezclar dos mezclas de colores ayb es a * b.

Averigüe cuál es la cantidad mínima de humo que Harry puede obtener al mezclar todas las mezclas.

Entrada

Habrá una serie de casos de prueba en la entrada.

La primera línea de cada caso de prueba contendrá n, el número de mezclas, 1 <= n <= 100.

La segunda línea contendrá n enteros entre 0 y 99, los colores iniciales de las mezclas.

Salida

Para cada caso de prueba, produzca la cantidad mínima de humo.

Ejemplo

```
Entrada:
2
18 19
3
40 60 20

Salida:
342
2400
```

En el segundo caso de prueba, hay dos posibilidades:

- primero mezcle 40 y 60 (humo: 2400), obteniendo 0, luego mezcle 0 y 20 (humo: 0); la cantidad total de humo es 2400
- primero mezcle 60 y 20 (humo: 1200), obteniendo 80, luego mezcle 40 y 80 (humo: 3200); la cantidad total de humo es 4400

El primer escenario es una forma mucho mejor de proceder.

Enviar solución!

Añadido por:	Tomek Czajka
Fecha:	2005-05-03
Límite de tiempo:	3s
Límite de fuente:	50000B
Limite de memoria:	1536MB
Racimo:	Cubo (Intel G860)
Idiomas:	Todos excepto: NODEJS PERL6 VB.NET
Recurso:	Concurso de Programación Purdue Entrenamiento

Requisitos de voto

- ser usuario de spoj durante al menos 5 días
- Resuelto 1 de 15 problemas necesarios.
- resuelve este problema

Etiquetas propias

Tag name — Añadir

Sphere online judge — PROBLEMS · STATUS · RANKS · DISCUSS · CONTESTS · PROFILE

judge status

< Previous 1 2 3 4 5 Next >

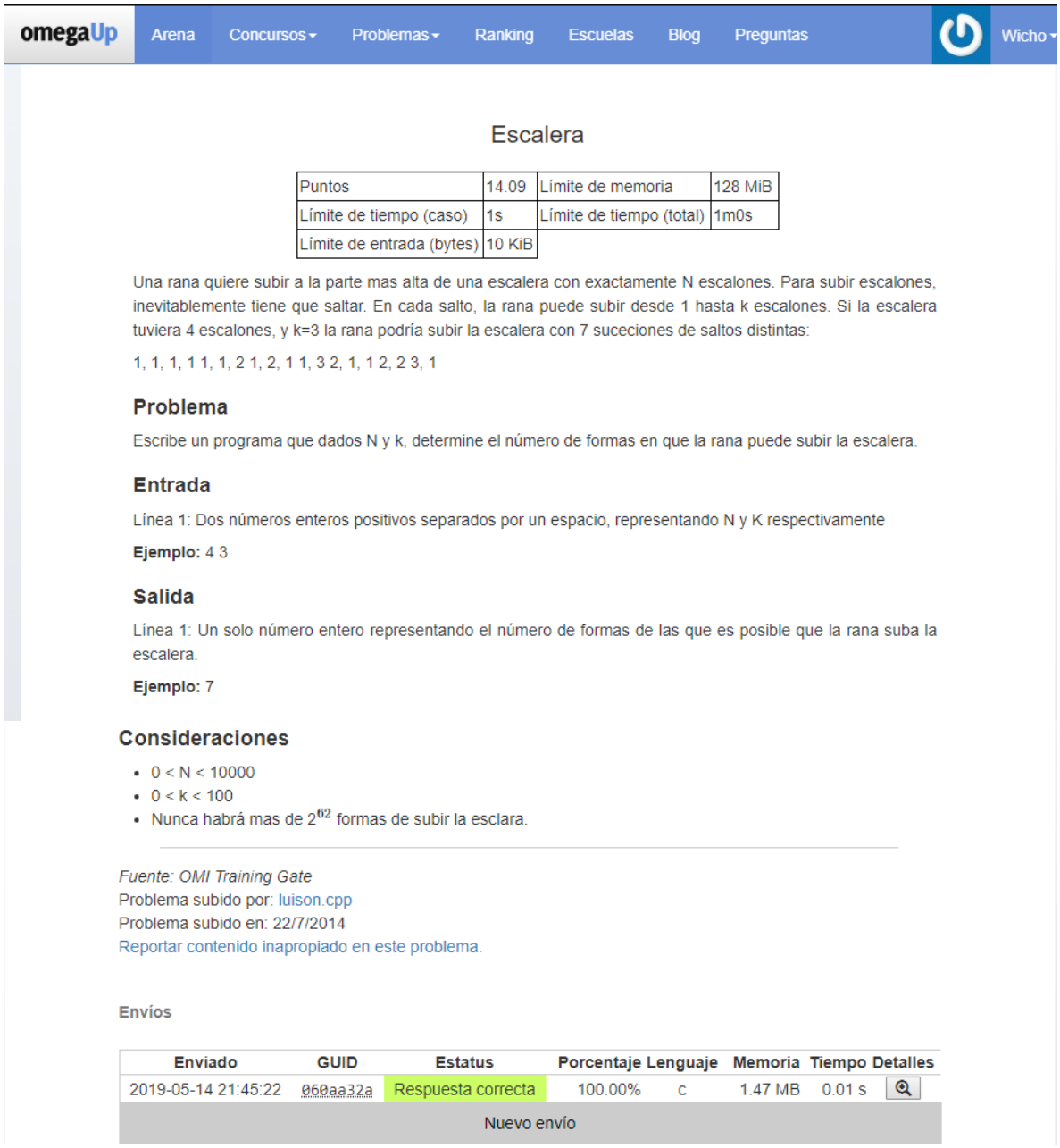
ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
23765309	2019-05-15 04:12:32	wicho	Mixtures	accepted edit ideone it	0.00	16M	CPP

El problema nos presenta de una manera amigable para entenderlo a Harry Potter que tiene varias mezclas y quiere mezclar todas y cada que las junta sale un color a + b y el humo es a* b, la entrada tiene que ser el número de mezclas y enteros de los colores iniciales de las mezclas. La salida para cada caso será la cantidad mínima de humo.

Diseño de solución Bottom Up, simplemente se hacen los casos de prueba; para cuando se tienen las 2 posibilidades del humo. Entonces se saca el porcentaje de las mezclas y humo y retorna el mínimo encontrado.

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

#define MAX 100+10
int n,valores[MAX],m[MAX][MAX];

int mix(int l,int r){
    if(m[l][r] != -1)
        return m[l][r];
    if(l == r)
        return 0;
    if(l+1 == r)
        return (valores[l]-valores[l-1])*(valores[r]-valores[r-1]);

    int res = 1<<20;
    for(int i=l; i<r; i++){
        int tmp=((valores[i]-valores[l-1])%100)*((valores[r]-valores[i])%100
        res=min(res,tmp+mix(l,i)+mix(i+1,r));
    }
    m[l][r]=res;
    return res;
}
int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);
    while(cin>>n){
        for(int i=0; i<MAX; i++)
            for(int j=0; j<MAX; j++)
                m[i][j]=-1;

        valores[0]=0;
        for(int i=1; i<=n; i++){
            cin>>valores[i];
            valores[i] += valores[i-1];
        }
        cout<<mix(1,n)<< "\n";
    }
}
```

VI. ESCALERAS



omegaUp Arena Concursos Problemas Ranking Escuelas Blog Preguntas Wicho

Escalera

Puntos	14.09	Límite de memoria	128 MiB
Límite de tiempo (caso)	1s	Límite de tiempo (total)	1m0s
Límite de entrada (bytes)	10 KiB		

Una rana quiere subir a la parte mas alta de una escalera con exactamente N escalones. Para subir escalones, inevitablemente tiene que saltar. En cada salto, la rana puede subir desde 1 hasta k escalones. Si la escalera tuviera 4 escalones, y k=3 la rana podría subir la escalera con 7 sucesiones de saltos distintas:

1, 1, 1, 1 1, 1, 2 1, 2, 1 1, 3 2, 1, 1 2, 2 3, 1

Problema

Escribe un programa que dados N y k, determine el número de formas en que la rana puede subir la escalera.

Entrada

Línea 1: Dos números enteros positivos separados por un espacio, representando N y K respectivamente

Ejemplo: 4 3

Salida

Línea 1: Un solo número entero representando el número de formas de las que es posible que la rana suba la escalera.

Ejemplo: 7

Consideraciones

- 0 < N < 10000
- 0 < k < 100
- Nunca habrá mas de 2^{62} formas de subir la esclara.

Fuente: OMI Training Gate
Problema subido por: luison.cpp
Problema subido en: 22/7/2014
Reportar contenido inapropiado en este problema.

Envíos

Enviado	GUID	Estatus	Porcentaje	Lenguaje	Memoria	Tiempo	Detalles
2019-05-14 21:45:22	060aa32a	Respuesta correcta	100.00%	c	1.47 MB	0.01 s	

Nuevo envío

El problema simplemente nos pide que subamos a la parte más alta de una escalera saltando de 1 hasta k escalones en una escalera de n escalones, por lo que habría que hacer las combinaciones de saltos.

Para la solución simplemente es una función de recursividad donde se va sumando en el arreglo lo que devuelva la función mandando como argumentos el primer arreglo de n escalones menos el recorrido de la pocición en donde se encuentra el arreglo, los escalones que va a saltar y el arreglo de los saltos. La complejidad es de O(n).

```c
#include <stdio.h>
#include <string.h>

long saltos(long n, long m,long s[]){
    int i;
    if(n<0)
        return 0;
    if(n==0)
        return 1;
    if(s[n]==0){
        for(i=1;i<=m;i++){
            s[n]+=saltos(n-i,m,s);
        }
    }
    return s[n];
}

int main(){
    long n,k;
    scanf("%ld %ld",&n, &k);
    long s[n+1];
    memset(s,0,sizeof(long)*(n+1));
    long result=saltos(n,k,s);
    printf("%ld",result);
}
```