Monedas

Baq tiene n monedas, cada una con un valor m_i. A Baq no le gusta que las máquinas donde compra los billetes de metro le devuelvan cambio, por eso quiere saber si con sus n monedas puede pagar con exactitud cualquier cantidad.

Cual es la mínima cantidad de dinero que Baq no puede pagar con exactitud con sus n monedas?

Input Format

La entrada consiste en un entero t seguido de t casos.

Cada caso empieza con un entero n seguido de los n valores de las monedas, m_i.

Constraints

$1 \leq t \leq 500$

$1 \leq m_i \leq 10^9$

$1 \leq n$

13 Puntos $n \leq 10$

26 Puntos $n \leq 100$

30 Puntos $n \leq 1000$

31 Puntos $n \leq 10000$

Output Format

Una línea para cada caso con la mínima cantidad de dinero que Baq no puede pagar con exactitud.

Sample Input 0

```
3
5
2 4 16 8 1
3
1 2 3
4
3 5 7 1
```

Sample Output 0

```
32
7
2
```

Solución

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

using namespace std;

int main(){
    int t;
    cin >> t;
    for (int i = 0; i < t; i++){
        int n;
        long long int m;
        cin >> n;
        vector < long long int> M(n);

        for (int j = 0; j < n; j++){
            cin >> m;
            M[j] = m;

        }
        sort(M.begin(), M.end());
        long long int sm = 0;
        bool b = true;
        for(int j = 0; j < n and b; j++){
            if (M[j] -1 > sm) b = false;
            else sm += M[j];
        }
        cout << sm + 1 << endl;
    }
}
```

// Autor de la solución: Joan Hernanz (Concursante)

Montañas rusas

Hermenegilda está entrenando para concursar en la Olimpiada Internacional de Informática, pero no le queda mucho tiempo. Tras sopesar un rato qué estategia debería seguir, ha decidido que lo mejor que puede hacer es dejar de estudiar y centrar sus esfuerzos en convertirse en una concursante rusa, puesto que de esta forma seguro que obtendrá medalla. Para lograr este fin, viaja hasta el Parque de Atracciones Federal de Kazán, famoso por sus auténticamente rusas montañas rusas.

Una montaña rusa auténticamente rusa es una atracción que consta de $2l$ tramos, puestos uno detrás del otro. Solamente existen dos tipos de tramos: los ascendentes y los descendentes. Cada tramo mide exactamente una unidad rusa de longitud en su componente horizontal y una unidad rusa de longitud en su componente vertical. Es decir, que cada tramo avanza una unidad rusa de longitud ascendente o descendente y una unidad rusa de longitud horizontalmente, en función de si es un tramo ascendente o descendente. Además, las montañas rusas auténticamente rusas no tienen ningún tramo bajo tierra, y empiezan y acaban al nivel del suelo. Sorprendentemente, el Parque de Atracciones Federal de Kazán contiene todas las posibles montañas rusas auténticamente rusas.

En su empeño por convertirse en una concursante rusa, Hermenegilda empieza su entrenamiento metódicamente: quiere montarse en todas las montañas rusas auténticamente rusas que tengan una longitud de $2l$ unidades rusas de longitud y una altura de exactamente h unidades rusas de longitud; esto es, que su punto más elevado esté h unidades rusas de longitud por encima del nivel del suelo. Te pedimos que digas en cuantas montañas rusas auténticamente rusas deberá montarse Hermenegilda.

Input Format

La primera línea contiene un único entero positivo t, el número total de casos. Siguen t líneas, una para cada caso, cada una de ellas formada por dos enteros positivos: l y h.

Constraints

$1 \leq h, l \leq 9, 1 \leq t \leq 25$	20 Puntos
$1 \leq h, l \leq 13, 1 \leq t \leq 9$	20 Puntos
$1 \leq h, l \leq 200, 1 \leq t \leq 40000$	60 Puntos

Output Format

Para cada caso debes imprimir un único entero: el número de atracciones de longitud $2l$ y altura h en las que se montará Hermenegilda. Como esta cantidad puede ser muy elevada, te pedimos que la imprimas módulo 1000000007.

Sample Input 0

```
4
2 2
2 3
3 2
15 5
```

Sample Output 0

```
1
0
3
2665884
```

Solución

```
#include<iostream>
#include<vector>

using namespace std;

typedef long long int ll;

ll l, h;

ll mod = 1000000007;
vector<vector<vector<ll> > > a;

ll rec(ll cl, ll ch, ll mh) {
    ll sol;

    if(a[cl][ch][mh] == -1) {
        sol = 0;
        if(cl == 0) return 0;
        //DESC
        if(ch < mh) {
            sol += rec(cl-1, ch+1, mh);
        }
        //ASC
        if(ch > 0) {
        if(ch == mh) {
            sol += rec(cl-1, ch-1, mh-1);
        }
        sol += rec(cl-1, ch-1, mh);
        }
        sol = sol % mod;
        a[cl][ch][mh] = sol;
        return sol;


    }
    else {
        return a[cl][ch][mh];
    }
}


void caso() {
    cin >> l >> h;
    rec(2*l, 0, h);
    cout << a[2*l][0][h] << endl;
}


int main() {
        ios::sync_with_stdio(0);
        cin.tie(0);
```

```
     int t;
    cin >> t;
    a = vector<vector<vector<ll>>>(401, vector<vector<ll>>(201, vector<ll>(201, -1)));
    a[0][0][0] = 1;
    for(int i=0; i < t; i++) {
        caso();
    }

}
```

// Autor: Félix Moreno (Concursante)

Decodificando permutaciones

A los hermanos Baq y Babaq les gusta coleccionar permutaciones. La colección de Babaq es más grande que la de Baq (lleva más tiempo coleccionándolas). Por este motivo, Baq quiere quitarle algunas permutaciones a su hermano, pero tiene un pequeño problema: Babaq ha codificado sus permutaciones para que nadie se las pueda quitar.

La codificación se ha hecho de la siguiente manera: Si p es una permutación de los números de 1 a n, la codificación de Babaq es un vector de n enteros c que cumplen que c_i (el i-ésimo número de c) es el número de enteros menores que p_i que aparecen antes que p_i en la permutación. Formalmente, c_i = # de enteros j, tal que $j < i$ y $p_j < p_i$.

Ayudad a Baq a decodificar las permutaciones de su hermano.

Input Format

Cada caso empieza con un entero t, el número de permutaciones que Baq quiere decodificar. Le siguen la descripción de las t permutaciones a decodificar. Cada codificación viene descrita por un entero n, el número de elementos de la permutación, y n enteros c_i que describen la codificación.

Output Format

Para cada caso escribid t líneas. En la t-ésima, escribid la t-ésima permutación decodificada.

$1 \leq n \leq 8, t = 36$ (21 puntos)
$1 \leq n \leq 2000, t = 40$ (37 puntos)
$1 \leq n \leq 10^5, t = 25$ (42 puntos)

Sample Input 0

```
2
3
0 0 2
5
0 1 2 0 2
```

Sample Output 0

```
2 1 3
2 4 5 1 3
```

Solución de 100 puntos

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> tree;

int ts;
```

```
void update(int x, int val){
  tree[x] = val;
  x /= 2;
  while (x > 0){
    tree[x] = tree[2*x]+tree[2*x+1];
    x /= 2;
  }
}

int query(int x, int l, int r, int k){
  if (r - l == 1) return x - ts;
  if (k >= tree[2*x]){
    return query(2*x+1, (l+r)/2, r, k - tree[2*x]);
  }
  else{
    return query(2*x, l, (l+r)/2, k);
  }
}

int main(){
  int t;
  cin >> t;
  while(t--){
    int n;
    cin >> n;
    vector<int> c(n);
    ts = 1;
    while (ts < n) ts <<= 1;
    tree = vector<int> (ts*2, 0);
    for (int i = 0; i < n; ++i) update(ts+i, 1);
    for (int i = 0; i < n; ++i) cin >> c[i];
    vector<int> res(n);
    for (int j = n-1; j >= 0; --j){
      res[j] = query(1, 0, ts, c[j])+1;
      update(ts+res[j]-1, 0);
    }
    for (int i = 0; i < n; ++i){
      cout << res[i] << " ";
    }
    cout << endl;
  }
}
```

// Autor: Miquel Ortega (Miembro del comité organizador)

Solución de 58 puntos

```
#include <cmath>
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
```

```cpp
int main() {
	int t;
	cin >> t;
	for (int t0 = 0; t0<t; t0++) {
		int n;
		cin >> n;
		vector<int> nums;
		vector<int> res;
		for (int i = 0; i<n; i++) {
			int v;
			cin >> v;
			nums.push_back(v);
			res.push_back(0);
		}
		res[0] = 1;
		int lastval = 0;
		for (int i = 1; i<n; i++) {
			res[i] = nums[i] + 1;
			for (int k = 0; k<i; k++) {
				if (res[k] >= res[i])res[k]++;
			}
		}
		cout << res[0];
		for (int d = 1; d < n; d++) {
			cout << " " << res[d];
		}
		cout << endl;
	}
	return 0;
}
```

// Autor: Albert Martín (Concursante)

Girona temps de flors

Desde hace más de 60 años la ciudad de Girona celebra la exposición de flores "Girona temps de flors", en la que se decora cada rincón y espacio emblemático del barrio antiguo con flores. El resultado es una ciudad llena de flores y, como no, turistas curiosos de todo el mundo que llenan las calles del barrio antiguo de Girona para disfrutar de la maravillosa exposición.

Pero como todo, tiene una parte oscura. Los pobres ciudadanos que viven tranquilamente durante todo el año en el barrio antiguo, ven sus calles infestadas de turistas que parece que no saben distinguir entre una carretera y una acera, o entre una flor y un coche que quiere pasar.

Después de años viviendo en el barrio antiguo de Girona, este año Cesc va a cumplir el sueño secreto de todos los residentes de su barrio: ¡empujar al máximo número de turistas en el camino a casa!

Este año la exposición consta de n puntos decorados con flores, en el punto i hay t_i turistas haciendo fotos. El barrio antiguo tiene una gran cantidad de callejones y entre cada dos puntos de exposición de flores existe un callejón que los une. El callejón que va del punto i al j tiene longitud a_{ij}, tened en cuenta que los callejones son de un solo sentido.

Cesc empieza en el punto 0 y quiere ir a su casa que está en el punto 1 empujando el máximo número de turistas posible. Pero tiene un poco de prisa y no quiere tardar más de S segundos, si cada segundo recorre una unidad de distancia y puede empujar a los turistas sin perder tiempo, ¿cuantos turistas puede empujar de camino a casa como máximo?

Importante: Para ir del punto i al punto j no hace falta usar el callejón directo, se puede usar cualquier combinación de callejones.

Solo se puede empujar una vez a cada turista.

Se garantiza que siempre tiene tiempo de ir del punto 0 al punto 1.

Input Format

La entrada consiste en varios casos.

Cada caso empieza con los enteros n, S.

La siguiente línea contiene los n enteros t_i.

Cada una de las siguientes n líneas contiene los enteros a_{ij}. La línea i contiene los enteros $a_{i0}, a_{i1}, \ldots, a_{in}$.

Constraints

$1 \leq a_{ij}, t_i \leq 1000$

$a_{ii} = 0$

$1 \leq S \leq 20000$

11 puntos $2 \leq n \leq 8$ y $a_{ij} = 1$ para todo i y j distintos.

12 puntos $2 \leq n \leq 8$ y a_{ij} es el tiempo mínimo para ir de i a j.

13 puntos $2 \leq n \leq 8$

11 puntos $2 \leq n \leq 15$ y $a_{ij} = 1$ para todo i y j distintos.

12 puntos $2 \leq n \leq 15$ y a_{ij} es el tiempo mínimo para ir de i a j.

13 puntos $2 \leq n \leq 15$

28 puntos $2 \leq n \leq 18$

Output Format

Una línea para cada caso con la respuesta.

Sample Input 0

```
4 4
887 778 916 794
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
```

Sample Output 0

```
3375
```

Explanation 0

Todos los $a_{ij} = 1$

Sample Input 1

```
3 1379
650 422 363
0 887 778
916 0 794
336 387 0
3 454
173 737 212
0 28 691
60 0 751
601 541 0
```

Sample Output 1

```
1435
910
```

Explanation 1

Todos los a_{ij} son la mínima distancia entre i y j.

Sample Input 2

```
8 7246
171 997 282 306 926 85 328 337
0 384 887 778 916 794 336 387
493 0 650 422 363 28 691 60
764 927 0 541 427 173 737 212
369 568 430 0 783 531 863 124
68 136 930 803 0 23 59 70
168 394 457 12 43 0 230 374
422 920 785 538 199 325 0 316
371 414 527 92 981 957 874 0
```

Sample Output 2

```
3432
```

Explanation 2

Caso general.

Solución

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;
typedef long long ll;

int main() {
	int n, s;
	while (cin >> n >> s) {
		vector< vector<ll> > d(n, vector<ll>(n));
		vector<ll> t(n);
		for (int i = 0; i < n; ++i) cin >> t[i];
		for (int i = 0; i < n; ++i) {
			for (int j = 0; j < n; ++j) cin >> d[i][j];
		}
		for (int k = 0; k < n; ++k) {
			for (int i = 0; i < n; ++i) {
				for (int j = 0; j < n; ++j) {
				    if (d[i][k]+d[k][j] < d[i][j]) d[i][j] = d[i][k]+d[k][j];
				}
			}
		}
		vector< vector<ll> > a(1 << n, vector<ll>(n, 0));
		vector< vector<ll> > r(1 << n, vector<ll>(n, s + 1));
		a[1][0] = t[0];
		r[1][0] = 0;
		queue< pair<ll, ll> > q;
		q.push({1, 0});
		while (!q.empty()) {
			pair<ll, ll> k = q.front();
			q.pop();
			ll ta = a[k.first][k.second];
			ll tr = r[k.first][k.second];
			for (int i = 0; i < n; ++i) {
				if (!(k.first&(1 << i)) and i != k.second) {
					ll m = k.first|(1 << i);
					if (tr + d[k.second][i] < r[m][i]) {
						if (r[m][i] == s + 1) {
							q.push({m, i});
							a[m][i] = ta + t[i];
						}
```

```
                                        r[m][i] = tr + d[k.second][i];
                                    }
                                }
                            }
                        }
                        ll best = 0;
                        for (int i = 0; i < (1 << n); ++i) best = max(best, a[i][1]);
                        cout << best << endl;
                }
}
```

// Autor: Jan Olivetti (Miembro del comité organizador)