Universidade de São Paulo - São Carlos

Instituto de Ciências Matemáticas e de Computação

MFP Unicamp

Jun 24, 2023

Contest (1)

template.cpp — 17 lines

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

// No decorrer do notebook, podem aparecer alguns dos seguintes
    comandos, para diminuir a quantidade de caracteres e
    facilitar copiar o codigo (nao eh necessario copiar isso,
    apenas entender caso apareca algum assim la na frente)
#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

int main() {
  ios::sync_with_stdio(false); cin.tie(nullptr);
  int x;
  cin >> x;
  cout << x << "\n";
}
```

template.py — 8 lines

```
# ler string
x = input()
# ler int
x = int(input()) # le int
# ler array de string na mesma linha separado por espaco
l = input().split(' ')
# ler array de int na mesma linha
l = list(map(int, input().split()))
```

terminal.txt — 12 lines

```
# compilar com flags de warning e sanitizers
g++ sol.cpp -o sol -Wall -std=c++17 -fsanitize=address,
    undefined -g

# compilar com flag de otimizacao -O2
g++ sol.cpp -o sol -Wall -std=c++17 -O2

# testar o input no arquivo 1.in
./sol < 1.in

# comparar a sua saida com a do arquivo 1.out
./sol < 1.in > my_out
diff -w 1.out my_out
```

troubleshoot.txt — 46 lines

```
Antes de submeter:
Pense em casos de teste simples se o sample nao for suficiente.
Os limites de tempo estao proximos? Gere casos maximos.
O uso de memoria esta ok?
Algum possivel overflow?
Selecione o arquivo certo para submeter.

Wrong answer:
Esta limpando as variaveis entre casos de teste?
Seu algoritmo suporta todo o intervalo de input?
Leia o enunciado novamente.
Algum caso de borda nao foi testado?
O problema foi entendido corretamente?
Alguma variavel nao inicializada?
Algum overflow?
Confundiu N com M, i com j, etc.?
Tem certeza que seu algoritmo funciona?
Algum caso especial que voce nao pensou?
Tem certeza que as funcoes de STL funcionam como voce pensou?
Adicione alguns "asserts" e talvez submeta novamente;
Crie casos de teste novos.
Pense passo a passo como seu algoritmo funciona em um caso
    simples.
Pare de pensar nesse problema, tome um ar fresco, va pro coffee
    .
O formato de saida esta correto (incluindo espacos em branco)?
Reescreva sua solucao do comeco.

Runtime error:
Testou todos os casos de borda localmente?
Alguma variavel nao inicializada?
Algum acesso invalido a vetor?
Algum 'assert' que possa falhar?
Alguma divisao por 0 (ou mod 0)?
Alguma recursao infinita?
Ponteiros ou iteradores invalidos?
Usando muita memoria?
Compile com as flags de fsanitize (veja exemplo em terminal.txt
    ) e rode localmente novamente para varios casos de teste.

Time limit exceeded:
Algum loop infinito possivel?
Qual a complexidade do seu algoritmo?
Esta copiando muitos dados desnecessarios (passagem de
    parametro por valor/ponteiro/referencia)?
Quao grande e o input/output? Considere usar scanf ou usar a
    linha de "fastcin" do template em C++.

Memory limit exceeded:
Qual o maximo de memoria que seu algoritmo precisa?
Voce esta limpando todas as estruturas de dados entre casos de
    teste`?
```

STL (2)

Possui algoritmos e containers que podem ser utilizados.

2.1 Estruturas de dados

2.1.1 vector<T>

Um vetor dinâmico, pode ter seu tamanho alterado durante a execucao e tudo mais

```
vector<int> v; // criar um vetor q armazena int
v.push_back(1); // insere no final. O(1)
v.pop_back(); // remove o cara do final. O(1)
v.insert(v.begin(), 1); // insere no comeco. O(n) - CUIDADO COM
    TLE!!
v.size(); // retorna a qtd de elementos no vetor

vector<long long> x(5, 1); // cria um vetor de 5 elementos
    valendo 1 cada
for (int i = 0; i < (int) x.size(); i++) cout << x[i]; //
    imprime 11111
for (int i : x) cout << i; // imprime 11111. Esse for e tipo um
    "for each"
```

2.1.2 string

Um vector de caracteres. Bem melhor do que manipular string em C

```
string x = "aba";
string y = "xxx";
string s = x + y; // concatena, s = abaxxx
cin >> s; // le do input
cout << s; // imprime na tela
x[1] = 'c'; // x = "aca" agora
```

2.1.3 pair<T1, T2>

Um par de elementos quaisquer. Ja possui o comparador de "<"implementado, facilitando a comparacao (considera o primeiro e desempata pelo segundo)

```
pair<int, double> p = {1, 3.5};
cout << p.first << " " << p.second; // 1 3.5
p.first = 10;
cout << p.first << " " << p.second; // 10 3.5

// pode ter pair de qqr coisa basicamente
pair<vector<int>, string> p2;
pair<pair<pair<int, int>, int>, int> p3; // tem q pegar p3.
    first.first.first
```

2.1.4 stack<T>

Uma estrutura de dados de pilha (FILO).

```
stack<int> st;
st.push(1);
st.push(2);
int old = st.top(); // old = 2
st.pop(); // tira da pilha
int cur = st.top(); // cur = 1
```

2.1.5 queue<T>

Uma estrutura de dados de fila (FIFO).

```
queue<int> qu;
qu.push(1);
qu.push(2);
int old = qu.front(); // old = 1
qu.pop(); // tira da pilha
int cur = qu.front(); // cur = 2
```

2.1.6 deque<T>

Double ended queue: uma fila dupla. Pode inserir e tirar do comeco ou do final.

```
deque<int> dq;
dq.push_back(3); // dq = {3}
dq.push_front(1); // dq = {1, 3}
dq.push_back(2); // dq = {1, 3, 2}
dq.back(); // retorna 2
```

```
dq.front(); // retorna 1
dp.pop_back(); // dq = {1, 3}
```

2.1.7 priority_queue<T>

Uma *max heap*. Faz operacoes de insercao e retornar o topo em O(logn).

```
priority_queue<int> pq;
pq.push(50); // pq = {50}
pq.push(10); // pq = {10, 50}
pq.top(); // retorna 50
pq.pop(); // pq = {10}
```

2.1.8 set<T>

Um conjunto matematico ordenado. Nao armazena valores repetidos. Utiliza uma arvore balanceada. Operacoes em O(logn).

```
set<int> st;
st.insert(10); // st = {10}
st.insert(1); // st = {1, 10}
st.insert(10); // st = {1, 10}
st.erase(10); // st = {1}
cout << *st.begin() << "\n"; // imprime 1

for (int x : st) cout << x << " "; // imprime todos os caras do
    set
```

2.1.9 multiset<T>

Um conjunto matematico ordenado. **Armazena** valores repetidos. Utiliza uma arvore balanceada. Operacoes em O(logn).

```
multiset<int> ms;
ms.insert(10); // ms = {10}
ms.insert(10); // ms = {10, 10}
ms.insert(1); // ms = {1, 10, 10}
ms.insert(1); // ms = {1, 1, 10, 10}
ms.erase(10); // remove TODOS os 10. ms = {1, 1}
ms.erase(ms.begin()); // remove so o cara do iterador que
    mandar. ms = {1}
cout << *ms.begin() << "\n"; // imprime 1

for (int x : ms) cout << x << " "; // imprime todos os caras do
    multiset
```

2.1.10 map<K, V>

Um dicionario "chave, valor". Pra cada chave, tem um valor associado. Caso vc acesse uma chave nao inicializada, e criado uma instância com valor (construtor vazio padrao).

```
map<string, int> mp;
mp["aba"] = 10;
mp["axxx"] = 11;
cout << mp["aba"]; // imprime 10
cout << mp["aaaaaaa"]; // imprime 0, inteiro padrao
mp.erase("aba"); // remove a chave "aba" e o valor q tava nela

for (auto p : mp) { // percorre todos os pair<key, value> do
    map
  cout << p.first << ": " p.second << " ";
} // nesse caso, imprimira "axxx: 11 aaaaaaa: 0 "
```

2.1.11 unordered

As estruturas `unordered_set<T>` e `unordered_map<K, V>` sao como os set e map, mas utilizam Hash para fazer as operacoes em O(1). Nao garantem ordem, e podem ser inclusive mais lentos que o set e map (podem ser criados casos de teste que "quebram"o hash, fazendo com que tenha muito conflito).

2.2 Iteradores

Um iterador e "como um ponteiro". Sao meio complicados de entender de primeira, mas sao muito utilizados pelos algoritmos da STL. Exemplos:

```
vector<int> v = {1, 2, 3};
vector<int>::iterator it; // tipo do iterador
// se fizer *it, pega o valor do cara daquele endereco

// exemplo
vector<int>::iterador ini = v.begin().
// E um iterador que aponta pro primeiro cara. No caso, *ini =
    1

auto ini = v.begin(); // tambem podemos utilizar a tipagem
    automatica "auto"
// pra nao precisar ficar escrevendo muita coisa

v.end(); // -> um elemento depois do ultimo cara. *v.end()
    daria erro
v.rbegin(); // iterador reverso (comeca do final pro comeco).
// no caso, v.rbegin() e o ultimo cara (*v.rbegin() = 3)
v.rend(); // iter. reverso, um cara antes do primeiro cara (*v.
    rend() da erro)
```

2.3 Algoritmos

2.3.1 sort(begin, end, <comparador>)

```
vector<int> v = {4, 2, 1};
sort(v.begin(), v.end(); // ordena em forma crescente
sort(v.begin(), v.end(), [](int a, int b) {
  return a > b;
}); // utiliza uma funcao lambda para ordenar de forma
    decrescente

// vc tambem pode fazer com uma funcao sem ser lambda
bool cmp(int a, int b) { return a > b; }
...
sort(v.begin(), v.end(), cmp);
```

2.3.2 lower_bound(begin, end, valor)

- **lower_bound:** retorna um iterador para o primeiro elemento *maior ou igual* ao valor enviado.
- **upper_bound:** retorna um iterador para o primeiro elemento *maior* que o valor enviado.

```
vector<int> v = {1, 2, 3, 3, 6}; // o vetor deve estar ordenado

auto it1 = lower_bound(v.begin(), v.end(), 1); // it pro indice
    0
auto it2 = upper_bound(v.begin(), v.end(), 1); // it pro indice
    1
auto it3 = lower_bound(v.begin(), v.end(), 3); // it pro indice
    2
auto it4 = upper_bound(v.begin(), v.end(), 3); // it pro indice
    4
auto it5 = lower_bound(v.begin(), v.end(), 6); // it pro indice
    4
auto it6 = upper_bound(v.begin(), v.end(), 6); // it do v.end()
auto it7 = lower_bound(v.begin(), v.end(), 7); // it do v.end()
auto it8 = lower_bound(v.begin(), v.end(), -1); // it pro
    indice 0
```

2.3.3 next_permutation(begin, end)

Gera permutacoes, muito util para problemas em que precisa testar algo para todas as permutacoes em $O(n!)$

```
vector<int> v = {3, 4, 1};
sort(v.begin(), v.end()); // ordena para ficar na "permutacao
    inicial"
do {
  for (int i : v) cout << i << " ";
  cout << "\n";
} while(next_permutation(v.begin(), v.end())); // imprime todas
    as 6 permutacoes do vetor v
```

2.3.4 min(val1, val2), max(val1, val2)

Retorna o valor minimo para os valores val1 e val2. Eles devem ser do mesmo tipo, e este tipo deve ter o operador < definido.

```
int a = 4, b = 10;
long long x = 1321, y = 923;

min(a, b); // 4
max(a, b); // 10
min(x, y); // 932
min(a, x); // ERRO, nao sao do mesmo tipo
min( (long long) a, x); // 4, cast antes para long long
min(1LL * a, x); // 4, multiplicacao de long long com INT gera
    long long
```

Algorithms (3)

DivideAndConquer.h

Description: Divide and Conquer pseudocode
Time: $\mathcal{O}(n \log n)$

f26c5b, 16 lines

```
// Estrutura basica (pseudocodigo)
int solve(vector<int> a) {
   int n = a.size();
   if (n == 1) return solucao(a);

   // divide
   vector<int> vl = a[0..n/2];
   vector<int> vr = a[n/2..n];

   // pega a resposta pra cada metade
   solucao_left = solve(vl);
   solucao_right = solve(vr);

   // "conquista"
   return merge(solucao_left, solucao_right);
}
```

BinarySearch.h

Description: Binary search on the answer
Time: $\mathcal{O}(\log(hi - lo))$

d65547, 44 lines

```
bool check(int k) { // int ou double
```

```
  // check se pode realizar a operacao com K
}

// dada uma funcao que retorna 0 0 0 0 0 1 1 1 1..
// retorna a posicao que ocorre o primeiro 1
int binary_search1(int n) {
  // trocar o LO e HI por valores limite
  int lo = 0, hi = n - 1, mi;
  while(lo < hi) {
    mi = (lo + hi) / 2;
    if (check(mi)) hi = mi;
    else lo = mi + 1;
  }
  return lo;
}

// dada uma funcao que retorna 1 1 1 1 1 0 0 0 0...
// retorna a posicao que ocorre o ultimo 1
int binary_search2(int n) {
  // trocar o LO e HI por valores limite
  int lo = 0, hi = n - 1, mi;
  while(lo < hi) {
    mi = (lo + hi + 1) / 2;
    if (check(mi)) lo = mi;
    else hi = mi - 1;
  }
  return lo;
}

// codigo para double
double binary_search() {
  const double EPS = 1e-9; // limiar de erro aceitavel

  double lo = 0, hi = 1e9, mi;
  while(hi - lo > EPS) {
    mi = (lo + hi) / 2;
    // nao esqueca de trocar tipo do check para double
    if (check(mi)) lo = mi;
    else hi = mi;
  }
  return lo;
}
```

TernarySearch.h

Description: Ternary search on a function. Get the minimum or maximum of a parabola, for example.

Time: $\mathcal{O}(\log(hi - lo))$ 9bda2b, 40 lines

```
const double EPS = 1e-9;

// function f() to be defined. Can be int as well
double f(double x) { ... }
double maximum(double lo, double hi) {
    while (hi - lo > EPS) {
        double m1 = lo + (hi - lo) / 3.0;
        double m2 = lo + 2.0 * (hi - lo) / 3.0;
        if (f(m1) < f(m2)) lo = m1;
        else hi = m2;
    }
    return (lo + hi) / 2;
}
double minimum(double lo, double hi) {
    while (hi - lo > EPS) {
        double m1 = lo + (hi - lo) / 3.0;
        double m2 = lo + 2.0 * (hi - lo) / 3.0;
        if (f(m1) < f(m2)) hi = m2;
        else lo = m1;
    }
    return (lo + hi) / 2.0;
}
// F(x) has to be strictly incr. or decr. where x is not max
int maximum(int lo, int hi) {
    while (lo < hi) {
        int mi = (lo + hi) / 2;
        if (f(mi) <= f(mi + 1)) lo = mi + 1;
        else hi = mi;
    }
    return lo;
}
// F(x) has to be strictly incr. or decr. where x is not min
int minimum(int lo, int hi) {
    while (lo < hi) {
        int mi = (lo + hi) / 2;
        if (f(mi) <= f(mi + 1)) hi = mi;
        else lo = mi + 1;
    }
    return lo;
}
```

Graph (4)

4.1 Fundamentals

4.1.1 Como representar

Em C++, podemos representar uma lista de adjacencias como um vector de vectors, ou um array de vectors:

```
int n, m; cin >> n >> m; // n=vertices, m=arestas
vector<vector<int>> edges(n); // edges[i] = {adjacentes a i}

for (int i = 0; i < m; i++) {
   int u, v; cin >> u >> v;
   edges[u].push_back(v); // u->v
   edges[v].push_back(u); // v->u(para grafos nao direcionados)
}

// tb poderia fazer um dos arrays estaticos:
const int MAXN = 1e5 + 5; // falando que o maximo de N eh 10^5
vector<int> edges[MAXN];
```

DFS.h

Description: Busca em profundidade, com implementacao recursiva.

Time: $\mathcal{O}(N + M)$ e4de52, 11 lines

```
const int MAXN = 1e5 + 5;

vector<int> edges[MAXN];
bool vis[MAXN];

void dfs(int u) {
   vis[u] = true;
   for (int v : edges[u]) if (!vis[v]) {
      dfs(v);
   }
}
```

BFS.h

Description: Busca em largura, com implementacao com fila.

Time: $\mathcal{O}(N + M)$ e4992f, 18 lines

```
const int MAXN = 1e5 + 5;

vector<int> edges[MAXN];
int dist[MAXN];

void bfs(int s) { // vertice para iniciar a busca
   dist[s] = 1;
   queue<int> qu;
   qu.emplace(s);
   while(!qu.empty()) {
      int u = qu.front();
      qu.pop();
      for (int v : edges[u]) if (!dist[u]) {
         dist[v] = dist[u] + 1;
         qu.emplace(v);
      }
   }
}
```

Dijkstra.h

Description: Caminho mínimo a partir de um vértice para todos os outros em um grafo ponderado.

Time: $\mathcal{O}((N + M) \log n)$ 12544c, 22 lines

```
const int MAXN = 1e5 + 5;

vector<pair<int, int>> edges[MAXN]; // edges[u] = {pair(v, w)}
ll dist[MAXN];

void dijkstra(int s) { // vertice para iniciar a busca
   memset(dist, 0x3f, sizeof dist); // inicializa todo mundo
        com INF

   dist[s] = 0;
   priority_queue<pair<ll, int>> pq;
   pq.emplace(0, 0);
   while(!pq.empty()) {
      auto [d, u] = pq.top();
      pq.pop();
      if (-d > dist[u]) continue;

      for (auto [v, w] : edges[u]) if (dist[v] > dist[u] + w) {
         dist[v] = dist[u] + w;
         pq.emplace(-dist[v], v);
      }
   }
}
```

FloydWarshall.h

Description: Calcula a distsancia minima entre todos os pares de vertices em um grafo direcionado e ponderado (suporta tambem arestas negativas). A matriz de input m possui o valor das arestas i, j em $m[i][j]$, e $m[i][j] =$ inf se i e j nao sao adjacentes. Como output, $m[i][j]$ possui a distancia minima entre i e j, e inf se nao ha caminho.

Time: $\mathcal{O}(N^3)$ d7f3b5, 9 lines

```
ll m[MAXN][MAXN];

void floydWarshal() {
  for (int k = 0; k < n; k++)
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        if (m[i][j] > m[i][k] + m[k][j])
          m[i][j] = m[i][k] + m[k][j];
}
```

TopoSort.h

Description: Topological sorting. Given is an oriented graph. Output is an ordering of vertices, such that there are edges only from left to right. If there are cycles, the returned list will have size smaller than n – nodes reachable from cycles will not be returned.

Time: $\mathcal{O}(|V| + |E|)$ 66a137, 14 lines

```
vi topoSort(const vector<vi>& gr) {
  vi indeg(sz(gr)), ret;
```

```
  for (auto& li : gr) for (int x : li) indeg[x]++;
  queue<int> q; // use priority_queue for lexic. largest ans.
  rep(i,0,sz(gr)) if (indeg[i] == 0) q.push(i);
  while (!q.empty()) {
    int i = q.front(); // top() for priority queue
    ret.push_back(i);
    q.pop();
    for (int x : gr[i])
      if (--indeg[x] == 0) q.push(x);
  }
  return ret;
}
```

Mathematics (5)

5.1 Equations

$$ax^2 + bx + c = 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The extremum is given by $x = -b/2a$.

$$\begin{aligned} ax + by &= e \\ cx + dy &= f \end{aligned} \Rightarrow \begin{aligned} x &= \frac{ed - bf}{ad - bc} \\ y &= \frac{af - ec}{ad - bc} \end{aligned}$$

In general, given an equation $Ax = b$, the solution to a variable x_i is given by

$$x_i = \frac{\det A_i'}{\det A}$$

where A_i' is A with the i'th column replaced by b.

5.2 Recurrences

If $a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k}$, and $r_1, \ldots, r_k$ are distinct roots of $x^k + c_1 x^{k-1} + \cdots + c_k$, there are $d_1, \ldots, d_k$ s.t.

$$a_n = d_1 r_1^n + \cdots + d_k r_k^n.$$

Non-distinct roots r become polynomial factors, e.g. $a_n = (d_1 n + d_2) r^n$.

5.3 Trigonometry

$$\sin(v + w) = \sin v \cos w + \cos v \sin w$$
$$\cos(v + w) = \cos v \cos w - \sin v \sin w$$

$$\tan(v + w) = \frac{\tan v + \tan w}{1 - \tan v \tan w}$$
$$\sin v + \sin w = 2 \sin \frac{v + w}{2} \cos \frac{v - w}{2}$$
$$\cos v + \cos w = 2 \cos \frac{v + w}{2} \cos \frac{v - w}{2}$$
$$(V + W) \tan(v - w)/2 = (V - W) \tan(v + w)/2$$

where V, W are lengths of sides opposite angles v, w.

$$a \cos x + b \sin x = r \cos(x - \phi)$$
$$a \sin x + b \cos x = r \sin(x + \phi)$$

where $r = \sqrt{a^2 + b^2}, \phi = \operatorname{atan2}(b, a)$.

5.4 Geometry

5.4.1 Triangles

Side lengths: a, b, c

Semiperimeter: $p = \dfrac{a + b + c}{2}$

Area: $A = \sqrt{p(p - a)(p - b)(p - c)}$

Circumradius: $R = \dfrac{abc}{4A}$

Inradius: $r = \dfrac{A}{p}$

Length of median (divides triangle into two equal-area triangles): $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$

Length of bisector (divides angles in two):

$$s_a = \sqrt{bc\left[1 - \left(\frac{a}{b + c}\right)^2\right]}$$

Law of sines: $\dfrac{\sin \alpha}{a} = \dfrac{\sin \beta}{b} = \dfrac{\sin \gamma}{c} = \dfrac{1}{2R}$

Law of cosines: $a^2 = b^2 + c^2 - 2bc \cos \alpha$

Law of tangents: $\dfrac{a + b}{a - b} = \dfrac{\tan \frac{\alpha + \beta}{2}}{\tan \frac{\alpha - \beta}{2}}$

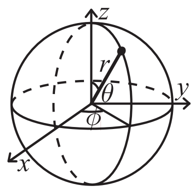
5.4.2 Quadrilaterals

With side lengths a, b, c, d, diagonals e, f, diagonals angle θ, area A and magic flux $F = b^2 + d^2 - a^2 - c^2$:

$$4A = 2ef \cdot \sin \theta = F \tan \theta = \sqrt{4e^2 f^2 - F^2}$$

For cyclic quadrilaterals the sum of opposite angles is $180°$, $ef = ac + bd$, and $A = \sqrt{(p - a)(p - b)(p - c)(p - d)}$.

5.4.3 Spherical coordinates



$$\begin{aligned} x &= r \sin \theta \cos \phi & r &= \sqrt{x^2 + y^2 + z^2} \\ y &= r \sin \theta \sin \phi & \theta &= \operatorname{acos}(z/\sqrt{x^2 + y^2 + z^2}) \\ z &= r \cos \theta & \phi &= \operatorname{atan2}(y, x) \end{aligned}$$

5.5 Derivatives/Integrals

$$\frac{d}{dx} \arcsin x = \frac{1}{\sqrt{1 - x^2}} \qquad \frac{d}{dx} \arccos x = -\frac{1}{\sqrt{1 - x^2}}$$
$$\frac{d}{dx} \tan x = 1 + \tan^2 x \qquad \frac{d}{dx} \arctan x = \frac{1}{1 + x^2}$$
$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int x \sin ax = \frac{\sin ax - ax \cos ax}{a^2}$$
$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2} \operatorname{erf}(x) \qquad \int x e^{ax} dx = \frac{e^{ax}}{a^2}(ax - 1)$$

Integration by parts:

$$\int_a^b f(x) g(x) dx = [F(x) g(x)]_a^b - \int_a^b F(x) g'(x) dx$$

5.6 Sums

$$c^a + c^{a+1} + \cdots + c^b = \frac{c^{b+1} - c^a}{c - 1}, c \neq 1$$

$$1 + 2 + 3 + \cdots + n = \frac{n(n + 1)}{2}$$
$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(2n + 1)(n + 1)}{6}$$
$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n + 1)^2}{4}$$
$$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n + 1)(2n + 1)(3n^2 + 3n - 1)}{30}$$

5.7 Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, \ (-\infty < x < \infty)$$

$$\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, \ (-1 < x \leq 1)$$

$$\sqrt{1 + x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, \ (-1 \leq x \leq 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, \ (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, \ (-\infty < x < \infty)$$

5.8 Probability theory

Let X be a discrete random variable with probability $p_X(x)$ of assuming the value x. It will then have an expected value (mean) $\mu = \mathbb{E}(X) = \sum_x x p_X(x)$ and variance $\sigma^2 = V(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2 = \sum_x (x - \mathbb{E}(X))^2 p_X(x)$ where σ is the standard deviation. If X is instead continuous it will have a probability density function $f_X(x)$ and the sums above will instead be integrals with $p_X(x)$ replaced by $f_X(x)$.

Expectation is linear:

$$\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$$

For independent X and Y,

$$V(aX + bY) = a^2V(X) + b^2V(Y).$$

5.8.1 Discrete distributions

Binomial distribution

The number of successes in n independent yes/no experiments, each which yields success with probability p is $\mathrm{Bin}(n, p)$, $n = 1, 2, \ldots$, $0 \le p \le 1$.

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\mu = np,\ \sigma^2 = np(1-p)$$

$\mathrm{Bin}(n, p)$ is approximately $\mathrm{Po}(np)$ for small p.

First success distribution

The number of trials needed to get the first success in independent yes/no experiments, each wich yields success with probability p is $\mathrm{Fs}(p)$, $0 \le p \le 1$.

$$p(k) = p(1-p)^{k-1},\ k = 1, 2, \ldots$$

$$\mu = \frac{1}{p},\ \sigma^2 = \frac{1-p}{p^2}$$

Poisson distribution

The number of events occurring in a fixed period of time t if these events occur with a known average rate κ and independently of the time since the last event is $\mathrm{Po}(\lambda)$, $\lambda = t\kappa$.

$$p(k) = e^{-\lambda}\frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots$$

$$\mu = \lambda,\ \sigma^2 = \lambda$$

5.8.2 Continuous distributions

Uniform distribution

If the probability density function is constant between a and b and 0 elsewhere it is $\mathrm{U}(a, b)$, $a < b$.

$$f(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

$$\mu = \frac{a+b}{2},\ \sigma^2 = \frac{(b-a)^2}{12}$$

Exponential distribution

The time between events in a Poisson process is $\mathrm{Exp}(\lambda)$, $\lambda > 0$.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \ge 0 \\ 0 & x < 0 \end{cases}$$

$$\mu = \frac{1}{\lambda},\ \sigma^2 = \frac{1}{\lambda^2}$$

Normal distribution

Most real random values with mean μ and variance σ^2 are well described by $\mathcal{N}(\mu, \sigma^2)$, $\sigma > 0$.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

If $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ then

$$aX_1 + bX_2 + c \sim \mathcal{N}(\mu_1 + \mu_2 + c, a^2\sigma_1^2 + b^2\sigma_2^2)$$

5.9 Markov chains

A *Markov chain* is a discrete random process with the property that the next state depends only on the current state. Let $X_1, X_2, \ldots$ be a sequence of random variables generated by the Markov process. Then there is a transition matrix $\mathbf{P} = (p_{ij})$, with $p_{ij} = \Pr(X_n = i | X_{n-1} = j)$, and $\mathbf{p}^{(n)} = \mathbf{P}^n\mathbf{p}^{(0)}$ is the probability distribution for X_n (i.e., $p_i^{(n)} = \Pr(X_n = i)$), where $\mathbf{p}^{(0)}$ is the initial distribution.

π is a stationary distribution if $\pi = \pi\mathbf{P}$. If the Markov chain is *irreducible* (it is possible to get to any state from any state), then $\pi_i = \frac{1}{\mathbb{E}(T_i)}$ where $\mathbb{E}(T_i)$ is the expected time between two visits in state i. π_j/π_i is the expected number of visits in state j between two visits in state i.

For a connected, undirected and non-bipartite graph, where the transition probability is uniform among all neighbors, π_i is proportional to node i's degree.

A Markov chain is *ergodic* if the asymptotic distribution is independent of the initial distribution. A finite Markov chain is ergodic iff it is irreducible and *aperiodic* (i.e., the gcd of cycle lengths is 1). $\lim_{k\to\infty} \mathbf{P}^k = \mathbf{1}\pi$.

A Markov chain is an A-chain if the states can be partitioned into two sets $\mathbf{A}$ and $\mathbf{G}$, such that all states in $\mathbf{A}$ are absorbing ($p_{ii} = 1$), and all states in $\mathbf{G}$ leads to an absorbing state in $\mathbf{A}$. The probability for absorption in state $i \in \mathbf{A}$, when the initial state is j, is $a_{ij} = p_{ij} + \sum_{k\in\mathbf{G}} a_{ik}p_{kj}$. The expected time until absorption, when the initial state is i, is $t_i = 1 + \sum_{k\in\mathbf{G}} p_{ki}t_k$.

Combinatorial (6)

6.1 Fatorial

n	1	2	3	4	5	6	7	8	9	10
$n!$	1	2	6	24	120	720	5040	40320	362880	3628800
n	11	12	13	14	15	16	17			
$n!$	4.0e7	4.8e8	6.2e9	8.7e10	1.3e12	2.1e13	3.6e14			
n	20	25	30	40	50	100	150	171		
$n!$	2e18	2e25	3e32	8e47	3e64	9e157	6e262	>DBL_MAX		

6.2 Números de Catalan

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!n!}$$

$$C_0 = 1,\ C_{n+1} = \frac{2(2n+1)}{n+2}C_n,\ C_{n+1} = \sum C_iC_{n-i}$$

$C_n = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, \ldots$

- caminhos monotonicos sub-diagonais em um grid $n \times n$.
- strings com n pares de parenteses corretamente aninhados.
- arvores binarias com $n+1$ folhas (0 ou 2 filhos).
- arvores ordenadas com $n+1$ vertices.
- quantas vezes um poligono convexo com $n+2$ lados pode ser cortado em triangulos ao conectar vertices com linhas retas.

gema

MFP UNICAMP

MARATONA FEMININA DE PROGRAMAÇÃO