2023

Theoretical Computer Science Cheat Sheet

Definitions	
$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)$ $\forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0$ $\forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim\limits_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a \rvert < \epsilon$, $\forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\left[n \atop k \right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{ n \atop k \right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle n \atop k \right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\left\langle n \atop k \right\rangle\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \qquad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \qquad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \qquad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \qquad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n}\frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \qquad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\binom{n}{k} = \binom{n}{n-k}$,

4. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, **5.** $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,

6. $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, **7.** $\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,

8. $\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$, **9.** $\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,

10. $\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** $\left\{ n \atop 1 \right\} = \left\{ n \atop n \right\} = 1$,

12. $\left\{ n \atop 2 \right\} = 2^{n-1} - 1$, **13.** $\left\{ n \atop k \right\} = k\left\{ n-1 \atop k \right\} + \left\{ n-1 \atop k-1 \right\}$,

14. $\left[n \atop 1 \right] = (n-1)!$, **15.** $\left[n \atop 2 \right] = (n-1)!H_{n-1}$, **16.** $\left[n \atop n \right] = 1$, **17.** $\left[n \atop k \right] \ge \left\{ n \atop k \right\}$,

18. $\left[n \atop k \right] = (n-1)\left[n-1 \atop k \right] + \left[n-1 \atop k-1 \right]$, **19.** $\left\{ n \atop n-1 \right\} = \left[n \atop n-1 \right] = \binom{n}{2}$, **20.** $\sum_{k=0}^{n}\left[n \atop k \right] = n!$, **21.** $C_n = \frac{1}{n+1}\binom{2n}{n}$,

22. $\left\langle n \atop 0 \right\rangle = \left\langle n \atop n-1 \right\rangle = 1$, **23.** $\left\langle n \atop k \right\rangle = \left\langle n \atop n-1-k \right\rangle$, **24.** $\left\langle n \atop k \right\rangle = (k+1)\left\langle n-1 \atop k \right\rangle + (n-k)\left\langle n-1 \atop k-1 \right\rangle$,

25. $\left\langle 0 \atop k \right\rangle = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases}$ **26.** $\left\langle n \atop 1 \right\rangle = 2^n - n - 1$, **27.** $\left\langle n \atop 2 \right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,

28. $x^n = \sum_{k=0}^{n}\left\langle n \atop k \right\rangle\binom{x+k}{n}$, **29.** $\left\langle n \atop m \right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$, **30.** $m!\left\{ n \atop m \right\} = \sum_{k=0}^{n}\left\langle n \atop k \right\rangle\binom{k}{n-m}$,

31. $\left\langle n \atop m \right\rangle = \sum_{k=0}^{n}\left\{ n \atop k \right\}\binom{n-k}{m}(-1)^{n-k-m}k!$, **32.** $\left\langle\!\left\langle n \atop 0 \right\rangle\!\right\rangle = 1$, **33.** $\left\langle\!\left\langle n \atop n \right\rangle\!\right\rangle = 0$ for $n \ne 0$,

34. $\left\langle\!\left\langle n \atop k \right\rangle\!\right\rangle = (k+1)\left\langle\!\left\langle n-1 \atop k \right\rangle\!\right\rangle + (2n-1-k)\left\langle\!\left\langle n-1 \atop k-1 \right\rangle\!\right\rangle$, **35.** $\sum_{k=0}^{n}\left\langle\!\left\langle n \atop k \right\rangle\!\right\rangle = \frac{(2n)^{\underline{n}}}{2^n}$,

36. $\left\{ x \atop x-n \right\} = \sum_{k=0}^{n}\left\langle\!\left\langle n \atop k \right\rangle\!\right\rangle\binom{x+n-1-k}{2n}$, **37.** $\left\{ n+1 \atop m+1 \right\} = \sum_{k}\binom{n}{k}\left\{ k \atop m \right\} = \sum_{k=0}^{n}\left\{ k \atop m \right\}(m+1)^{n-k}$,

Template

```
#include <bits/stdc++.h>
#define debug(a) cout << #a << ": " << a << endl
#define test() int t; cin >> t; while(t--)
#define all(a) a.begin(), a.end()
#define fillWith(a, b) memset(a, b, sizeof(a))
#define Mod 1000000007
#define F first                                    #define S second
#define pb push_back
#define goFast() ios::sync_with_stdio(0); cin.tie(0);
cout.tie(0)
#define files(x) freopen(x, "r", stdin)
typedef long ll;
typedef long double ld;
```

- **Fibonacci:**
  - Cassini's identity: $F_{n-1}F_{n+1} - F_n^2 = (-1)^n$
  - The "addition" rule : $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$
  - K=n : $F_{2n} = F_n(F_{n+1} + F_{n-1})$
  - For any positive integer k, F_{nk} is multiple of F_n
  - The inverse is true, if F_m is multiple of F_n then m is multiple of n.
  - GCD : $GCD(F_m, F_n) = F_{GCD(m,n)}$
  - $f(n) = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}}$
- **Sieve**:

```
 const int MAX = 1e8;
bool prime[MAX];
void sieveOfEratosthenes(int n){
    for (int p = 2; p * p <= n; p++){
        if (prime[p] == true){
            for (int i = p * p; i <= n; i += p)
            prime[i] = false;
        }
    }
}
fillWith(prime, true); //in main
```

- **Comination :**

```
int dp[100][100];
 int combination(int n, int r){
    // nCr
    if(n == r || r == 0) return 1;
    if(dp[n][r] != -1) return dp[n][r];
    return dp[n][r] = combination(n - 1, r - 1) + combination(n - 1, r);
 }
```

```
fillWith(dp, -1);
```

- **Grapeee's Combinations:**

```
const int mod = 1e9 + 7;
int fa[100100];
int mul(int x, int y){
    return (ll) x * y % mod;
}
int po(int x, int y){
    if (!y) return 1;
    if (y & 1) return mul(x, po(x, y - 1));
    int z = po(x, y / 2);
    return mul(z, z);
}
int inv(int x){
    return po(x, mod - 2);
}
int C(int x, int y){
    if (y > x) return 0;
    return mul(mul(fa[x], inv(fa[y])), inv(fa[x - y]));
}
    int main(){ goFast(); fa[0] = 1; for (int i = 1; i <= 1e5; i++) fa[i] = mul(fa[i - 1], i); }
```

---

segment tree:

```
const int MAXN = 1000100;
int n, m, t[4*MAXN];
int lazy[4*MAXN];
```

❖ Build :

```
void build(int v, int tl, int tr) {
        if (tl == tr) {
                t[v] = 0;
        } else {
                int tm = (tl + tr) / 2;
                build(v*2, tl, tm);
                build(v*2+1, tm+1, tr);
                t[v] = t[v*2] + t[v*2+1];
        }
}
```

❖ push :

```
void push(int v) {
        if (lazy[v]) {
                t[v*2] += lazy[v];
                t[v*2+1] += lazy[v];
                lazy[v*2] += lazy[v];
                lazy[v*2+1] += lazy[v];
```

```
            lazy[v] = 0;
        }
    }
```

- **Update :**

```
    void update(int v, int tl, int tr, int l, int r, int new_val) {
    // cout << v << ' ' << tl << ' ' << tr << ' ' << l << ' ' << r << endl;
        if (l > r)
            return;
        if (l == tl && tr == r) {
            t[v] += new_val;
            lazy[v] += new_val;
        } else {
            push(v);
            int tm = (tl + tr) / 2;
            update(v*2, tl, tm, l, min(r, tm), new_val);
            update(v*2+1, tm+1, tr, max(l, tm+1), r, new_val);
            t[v] = t[v*2] + t[v*2+1];
        }
    }
```

- **get :**

```
    int get(int v, int tl, int tr, int i) { //cout << v << ' ' << tl << ' ' << tr << ' ' << i << endl;
    //cout << t[v] << endl;
        if (tl == tr)
            return t[tl];
        push(v);
        int tm = (tl + tr) / 2;
        if (i > tm)
            return get(v*2+1, tm+1, tr, i);
        else
            return get(v*2, tl, tm, i);
    }
```

Find in strings

```
string str = "x";
size_t found = str.find("x");
if(found != string::npos) cout << "Found at index: " << found;
```

Ordered_set

```
#include <ext/pb_ds/assoc_container.hpp>
```

```
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
tree_order_statistics_node_update> indexed_set;
auto x = s.find_by_order(2);
cout << s.order_of_key(7) << "\n";
```

Graph :

```
int n, m, visited[200200];
vector<int> myGraph[200200];
```

- **add weighted edge :**

```
void addWeightedEdge(int u, int v, int w){
    myGraph[u].push_back({w,v});
    myGraph[v].push_back({w,u});
}
```

- **DFS :**

```
void dfs(int u){
    if (visited[u])
        return;
    visited[u] = 1;
    for (int v : myGraph[u])
        dfs(v);
}
```

- **BFS:**

```
void bfs(int u){
    queue<int> q;
    q.push(u);
    visited[u] = true;
    while(!q.empty()){
        int s = q.front();
        q.pop();
        for(int v : myGraph[s]){
            if(!visited[v]){
                visited[v] = true;
                q.push(v);
            }
        }
    }
}
```

- **BFS shortest path**

```
vector<int> BFSshortestPath(int start){
    queue<int> q;
    vector<int> distance(n + 1, 1e8);
```

```
        q.push(start);
        distance[start] = 0;
        while (!q.empty()){
                int parent = q.front();
                q.pop();
                for (int son : myGraph[parent]){
                        if (distance[son] > distance[parent] + 1){
                                distance[son] = distance[parent] + 1;
                                q.push(son);
                        }
                }
        }
        return distance;
}
```

❖ 01BFS :

```
vector<int> 01Bfs(int start){
        deque<int> q;
        vector<int> distance(n + 1, 1e8);
        q.push_front(start);
        distance[start] = 0;
        while (!q.empty()){
                int parent = q.front();
                q.pop_front();
                for (auto pairSon : MyGraph[parent]){
                        int son = pairSon.first;
                        int weight = pairSon.second;
                        if (distance[son] > distance[parent] + weight){
                                distance[son] = distance[parent] + weight;
                                if (weight == 0)
                                        q.push_front(son);
                                else
                                        q.push_back(son);
                        }
                }
        }
        return distance;
}
```

❖ Dijkstra :

```
vector<ll> Dijkstra(int source){
        priority_queue<pair<ll, int>> q;
        vector<ll> Dist(n + 1, 1e18);
        Dist[source] = 0;
        q.push({Dist[source], source});
        while (!q.empty()){
                pair<ll, int> Top = q.top();
```

```
                q.pop();
                int Parent = Top.second;
                ll DistParent = (-1) * Top.first;
                if (DistParent > Dist[Parent])
                        continue; // Very Important
                for (auto PairSon : MyGraph[Parent]){
                        int son = PairSon.second;
                        ll Weight = PairSon.first;
                        if (DistParent + Weight < Dist[son]){
                                Dist[son] = DistParent + Weight;
                                q.push({(-1) * Dist[son], son});
                        }
                }
        }
        return Dist;
}
```

❖ **Floyd :**

```
int Distance[555][555];
void Initiate(){
        for(int i = 1; i <= n; i++)
                for(int j = 1; j <= n; j++)
                        Distance[i][j] = 1e9;
        for(int i = 1; i <= n; i++)
                Distance[i][i] = 0;
}
void Floyd(){
        for(int k = 1; k <= n; k++)
                for(int i = 1; i <= n; i++)
                        for(int j = 1; j <= n; j++)
                                Distance[i][j] = min(Distance[i][j],Distance[i][k] + Distance[k][j]);
}
```

❖ **DSU:**

```
int parent[200200], sizee[200200], numberOfComponents = n;
int root(int x){
        if (x == parent[x])
                return x;
        return parent[x] = root(parent[x]);
}
void Union(int x, int y){
        int rx = root(x);
        int ry = root(y);
        if (rx != ry){
                parent[rx] = ry;
                sizee[ry] += sizee[rx];
                numberOfComponents--;
```

```
		}
	}

	for (int i = 1; i <= n; i++){ // In Main
		parent[i] = i;
		sizee[i] = 1;
	}
```

❖ Graph On Grid

```
	int N;
	bool vis[N][N];
	// direction vectors:
	int dr = {-1, 1, 0, 0};
	int dc = {0 , 0, 1, -1};
	void dfsGrid(int i, int j){
		if(vis[i][j]) return;
		vis[i][j] = true;
		for(int k = 0 ; k < 4 ; i++){
			int r = i + dr[k];
			int c = j + dc[k];
			if(r < 0 || c < 0 || r > N || c > N) continue;
			dfsGrid(r,c);
		}
	}
```

Built-in function

- __builtin_clz(x): the number of zeros at the beginning of the number
- __builtin_ctz(x): the number of zeros at the end of the number
- __builtin_popcount(x): the number of ones in the number
- __builtin_parity(x): the parity (even or odd) of the number of ones

Bit manipulation

the formula x | (1 << k) sets the kth bit of x to one

the formula x & ~(1 << k) sets the kth bit of x to zero

the formula x ^ (1 << k) inverts the kth bit of x

The formula x & (x−1) sets the last one bit of x to zero

the formula x & −x sets all the one bits to zero, except for the last one bit

The formula x | (x−1) inverts all the bits after the last one bit

a positive number x is a power of two exactly when x & (x−1) = 0

```
int gcd(int a, int b){
   if(a > b)swap(a,b);
   if(a == 0){
      return b;
   }
    return gcd(b%a,a);
}

int lcm(int a, int b){
   return (a*b)/gcd(a,b);
}
```