**CHEATSHEET WILLIAMANDREA**

**Djikstra Algorithm : O(n + m log m) //edge harus positif**

priority\_queue<pair<int,int>> q;

for (int i = 1; i <= n; i++) e[i] = 1e9;

e[x] = 0;

q.push({0,x});

while (!q.empty()) {

int a = q.top().second; q.pop();

if (z[a]) continue;

z[a] = 1;

for (auto b : v[a]) {

if (e[a]+b.second < e[b.ff]) {

e[b.ff] = e[a]+b.second;

q.push({-e[b.ff],b.ff});

}

}

}

**String to int**

string s="1234";

stringstream toint(s);

int x=0;

toint>>x;

**string tostring(int s){**

stringstream ss;

ss<<s;

return ss.str();

}

**Build Permutation (jika pakai length k, tambah parameter start)**

**void gen() {**

if (v.size() == n) { // process permutation v

} else {

for (int i = 1; i <= n; i++) {

if (p[i]) continue;

p[i] = 1;

v.push\_back(i);

gen();

p[i] = 0;

v.pop\_back();

}

}

}

**Sieve of Eratosthenes**

int a[100005];

for (int x = 2; x <= n; x++) {

if (a[x]) continue;

for (int u = 2\*x; u <= n; u += x) {

a[u] = x;

}

}

**Path in a Grid (Berjalan dari ujung kiri atas ke ujung kanan bawah ambil poin terbesar)**

**int f(int y,int x){**

if (y==1 && x==1) return r[1][1];

if (y==1) return f(1,x-1)+r[1][x];

if (x==1) return f(y-1,1)+r[y][1];

return max(f(y,x-1),f(y-1,x))+r[y][x];

} cout<<f(n,n)<<"\n";

**Maximum Subarray Sum (Mencari jumlah sum terbesar di dalam Array)**

int p=0,s=0;

for (int i=1; i<=n; i++){

s=max(x[i],s+x[i]);

p=max(p,s);

} cout<<p<<"\n";

**Prefix Sum (Index dimulai dari 1)**

p[0]=0;

for (int i=0; i<n; i++){

cin>>x[i];

p[i+1]=p[i]+x[i];

} cout<<p[r]-p[l-1]<<endl;

**Subarray Sum (Mencari index L sampai R dengan jumlah query h)**

int sum=x[1],l=1,r=1;

while (r<=h){

int nxt=sum+x[r+1];

if (nxt==h){

r++; cout<<l<<" "<<r<<"\n";

return 0;

}else if (nxt>h){

sum-=x[l];

l++;

}else if (nxt<h){

r++;

sum+=x[r];

}

} cout<<-1<<"\n";

Signed Binary to Decimal

int p=pow(2, s.size()-1), ans=0;

rep (i, 0, s.size()-1){

if (i==0 && s[i]=='1') ans=p\*-1;

else ans+=((s[i]-'0')\*p);

p/=2;

}

cout<<ans<<"\n";

**Coin Change Problem (Top Down)**

**int coinchange(int x){**

if (x==0) return 0;

else if (memo[x]!=0)return memo[x];

else{

int best=1000000000;

for (int k=0; k<m; k++){

if (a[k]<=x) best=min(best,solve(x-a[k])+1);

}

memo[x]=best;

return memo[x];

}

} cout<<solve(n)<<endl;

**Coin Change Problem : Menghitung kemungkinan pembentukan koin**

/\* x=jumlah koin yang ingin dibentuk \*/

d[0] = 1;

for (int i = 1; i <= x; i++) {

for (int j = 1; j <= k; j++) {

if (i-c[j] < 0) continue;

d[i] += d[i-c[j]];

}

}

**int knapsack(int posisi, int mx){**

if (posisi>n) return 0;

if (memo[posisi][mx]!=-1) return memo[posisi][mx];

int ans=knapsack(posisi+1,mx);

if (mx>=w[posisi]){

ans=max(ans,knapsack(posisi+1,mx-w[posisi])+v[posisi]);

}

memo[posisi][mx]=ans;

return ans;

} cout<<knapsack(0,m)<<"\n";

**int minStep(int n){**

if (n==1) return 0;

if (memo[n]!=-1) return memo[n];

int ans=1+minStep(n-1);

if (n%2==0) ans=min(ans,1+minStep(n/2));

if (n%3==0) ans=min(ans,1+minStep(n/3));

memo[n]=ans;

return ans;

} cout<<minStep(n)<<"\n";

**Generating Subset**

void gen(int k){

if (k==n+1) // process

else{

gen(k+1); v.push\_back(k);

gen(k+1); v.pop\_back();

}

}

**int LongestIncreasingSubsequence(int n){**

int lis[100001];

for (int i=0; i<n; i++){

lis[i]=1;

}

for (int i=1; i<n; i++){

for (int j=0; j<i; j++){

if (arr[i]>arr[j] && lis[i]<lis[j]+1) {

lis[i]=lis[j]+1;

}

}

}

int mx=0;

for (int i=0; i<n; i++){

if (i) mx=max(mx,lis[i]);

else mx=lis[i];

}

return mx;

}

**comparison**

bool comp(int a, int b){ //(const pii &i, const pii &j){

return a>b;

} sort(x,x+n,comp);

**Pembagian Big Number**

string bagi(string s, int dv){

string ans="";

int p=0, tmp=0;

while (p<=s.size()-1){

tmp=(tmp\*10)+(s[p++]-'0');

ans+=(tmp/dv)+'0';

tmp-=dv\*(tmp/dv);

}

if (ans=="") ans="0";

while (ans[0]=='0') ans.erase(0, 1);

return ans;

}

**Penjumlahan Big Number**

string tambah(string a, string b){

int lenA=a.size(), lenB=b.size();

if (lenA<lenB){

rep (i, 1, (lenB-lenA)) a='0'+a;

}else if (lenA>lenB){

rep (i, 1, (lenA-lenB)) b='0'+b;

}

int tmp=0;

string c="";

repp (i, a.size()-1, 0){

int A=a[i]-'0', B=b[i]-'0', C=A+B;

c=digit[(C%10)+tmp]+c;

tmp=C/10;

}

if (tmp!=0) c=digit[tmp]+c;

return c;

}

**Maximum Spanning Tree (Prim Algorithm)**

vector<pii> v[101];

priority\_queue<pii> q;

bool vis[101]={};

void proses(int nd){

vis[nd]=1;

rep (i, 0, (int)v[nd].size()-1){

pii p=v[nd][i];

if (!vis[p.ff]) q.push({p.ss, p.ff});

}

}

ll mst(int nd\_awal){

proses(nd\_awal);

ll ans=0;

while (!q.empty()){

pii f=q.top(); q.pop();

int w=f.ff;

int u=f.ss;

if (!vis[u]){

ans+=w;

proses(u);

}

}

return ans;

}

**Berjalan dari ujung kiri atas ke ujung kanan bawah dengan mengambil angka 1 seminimal mungkin dan dapat berjalan hanya 4 arah**

char c[1005][1005]={};

int mn[1005][1005]={};

bool valid(int py, int px){

return py>=1 && py<=n && px>=1 && px<=n;

}

void bfs(int y, int x){

priority\_queue<piii, vector<piii>, greater<piii> >q;

bool vis[1005][1005]={};

q.push({0, {y, x}});

vis[y][x]=1;

while (!q.empty()){

piii f=q.top(); q.pop();

rep (i, 0, 3){

int batu=f.ff;

int py=f.ss.ff+dy[i];

int px=f.ss.ss+dx[i];

if (valid(py, px) && !vis[py][px]){

vis[py][px]=1;

mn[py][px]=batu+(c[py][px]-'0');

q.push({batu+(c[py][px]-'0'), {py, px}});

}

}

}

} bfs(1, 1); cout<<mn[n][n]<<endl;