TSP.cpp

```cpp
//TSP
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MAX 10
#define INF 987654321

using namespace std;

int N,dist[MAX][MAX],cache[MAX][1<<MAX];

int TSP(int here, int visited){
  if(visited == (1<<N)-1) return dist[here][0];
  int& ret = cache[here][visited];
  if(ret != -1) return ret;
  ret = INF;
  for(int i=0;i<N;i++){
    if(dist[here][i]==0 || visited & (1<<i)) continue;
    ret = min(ret,dist[here][i] + TSP(i,visited+(1<<i)));
  }
  return ret;
}

int main(){
  memset(cache,-1,sizeof(cache));
  scanf("%d",&N);
  for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)
      scanf("%d",&dist[i][j]);
  printf("%d\n",TSP(0,1));
}
```

bfs.cpp

```cpp
//bfs와 연결리스트 구현
//노드의 수 간선의 수 입력 받아 간선의 수 만큼 시작노드 끝노드 입력
//이 알고리즘은 연결 요소의 개수를 구한다.
#include <stdio.h>
#include <vector>
#include <queue>

using namespace std;

vector <vector<int> > v(1005); //연결리스트
queue <int> q;

int arr[1000]={0},check=1; //체크리스트

void bfs(int num){
  q.push(num);
  arr[num]=1;
  while(!q.empty()){
```

```
    int x=q.front(); q.pop();
    for(int i=0;i<v[x].size();i++){
      if(arr[v[x][i]]==0){
        q.push(v[x][i]);
        arr[v[x][i]]=1;
      }
    }
  }
}

int main(){
  int N,M;
  scanf("%d %d",&N,&M); //노드의 수, 간선의 수
  for(int i=0;i<M;i++){
    int U,V;
    scanf("%d %d",&U,&V);
    v[U].push_back(V);
    v[V].push_back(U); //방향없는 그래프 이기 때문에.
  }
  for(int i=1;i<=N;i++){
    if(arr[i]==0){
      bfs(i);
      check++;
    }
  }
  printf("%d\n",check-1);
}
```

bino.cpp

```
//이항계수 dp
#include <cstdio>
#include <cstring>

int T,M,N,cache[30][30];

int bino(int n, int r){
  if(r==0 || n==r) return 1;
  if(cache[n][r] != -1) return cache[n][r];
  return cache[n][r] = bino(n-1,r-1)+bino(n-1,r);
}

int main(){
  memset(cache,-1,sizeof(cache));
  scanf("%d",&T);
  while(T--){
    scanf("%d %d",&N,&M);
    printf("%d\n",bino(M,N));
  }
}
```

bipartiteMatch.cpp

```cpp
#include <cstdio>
#include <vector>
#define MAX_N 201
#define MAX_M 201

using namespace std;

//A와 B의 정점의 개수
int N,M;
//adj[i][j]=Ai와 Bj가 연결되어 있는가?
bool adj[MAX_N][MAX_M];
//각 정점에 매칭된 상대 정점의 번호를 저장
vector<int> aMatch, bMatch;
//dfs()의 방문 여부
vector<bool> visited;
//A의 정점인 a에서 B의 매칭되지 않은 정점으로 가는 경로를 찾는다
bool dfs(int a){
  if(visited[a]) return false;
  visited[a] = true;
  for(int b=0;b<M;b++){
    if(adj[a][b]){
      // b가 매칭되어 있지 않다면 bMatch[b]에서 부터 시작해 증가 경로를 찾는다.
      if(bMatch[b]==-1 || dfs(bMatch[b])){
        // 증가 경로 발견!! a와 b를 매치시킨다.
        aMatch[a] = b;
        bMatch[b] = a;
        return true;
      }
    }
  }
  return false;
}

//aMatch, bMatch 배열을 계산하고 최대 매칭의 크기를 반환
int bipartiteMatch(){
  //처음에는 어떤 정점도 연결되어 있지 않다.
  aMatch = vector<int>(N,-1);
  bMatch = vector<int>(M,-1);
  int size = 0;
  for(int start = 0; start<N; start++){
    visited = vector<bool>(N,false);
    if(dfs(start)) size++;
  }
  return size;
}

int main(){
  scanf("%d %d",&N,&M);
  for(int i=0;i<N;i++){
    int S; scanf("%d",&S);
```

```
    for(int j=0;j<S;j++){
      int X; scanf("%d",&X);
      adj[i][X-1] = true;
    }
  }
  printf("%d\n",bipartiteMatch());
}
```

bits.cpp

```
// 비트마스크 응용
#include <cstdio>

int bitCount(int x){
  if(x == 0) return 0;
  return x % 2 + bitCount(x / 2);
}

int main(){
  int pizza, toppings;
  // 최소 원소 찾기
  int firstTopping = (toppings & -toppings);
  // 최소 원소 지우기
  toppings &= (toppings - 1);
  // 모든 부분 집합 순회
  for(int subset = pizza; subset; subset = ((subset-1) & pizza)){
    // subset은 pizza의 부분 집합
  }
}
```

dijkstra.cpp

```
// 다익스트라 알고리즘
#include <cstdio>
#include <queue>
#include <vector>
#include <algorithm>
#define MAX_V 20001
#define INF 987654321

using namespace std;

int V,E,S;
vector<pair<int,int> > adj[MAX_V];

vector<int> dijkstra(int src){
  vector<int> dist(V+1,INF);
  dist[src]=0;
  priority_queue<pair<int,int> > pq;
  pq.push(make_pair(0,src));
  while(!pq.empty()){
    int cost = -pq.top().first, here = pq.top().second;
    pq.pop();
    if(dist[here]<cost) continue;
    for(int i=0;i<adj[here].size();i++){
```

```
      int there = adj[here][i].first;
      int nextDist = cost + adj[here][i].second;
      if(dist[there] > nextDist){
        dist[there] = nextDist;
        pq.push(make_pair(-nextDist, there));
      }
    }
  }
  return dist;
}

int main(){
  scanf("%d %d %d",&V,&E,&S);
  for(int i=0;i<E;i++){
    int u,v,w;
    scanf("%d %d %d",&u,&v,&w);
    adj[u].push_back(make_pair(v,w));
  }
  vector<int> dist = dijkstra(S);
  for(int i=1;i<=V;i++){
    if(dist[i]==INF) printf("INF\n");
    else printf("%d\n",dist[i]);
  }
}
```

DisjointSet.cpp

```
// 상호 베타적 집합 자료구조
#include <cstdio>
#include <vector>
#include <algorithm>

using namespace std;

struct DisjointSet{
  vector<int> parent, rank;
  DisjointSet(int n) : parent(n), rank(n,1){
    for(int i=0;i<n;i++)
      parent[i]=i;
  }
  int find(int u){
    if(u == parent[u]) return u;
    return parent[u] = find(parent[u]);
  }
  void merge(int u, int v){
    u = find(u); v = find(v);
    if(u == v) return;
    if(rank[u] > rank[v]) swap(u,v);
    parent[u] = v;
    if(rank[u] == rank[v]) ++rank[v];
  }
};

```

```
int main(){
  int N,M;
  scanf("%d %d",&N,&M);
  DisjointSet ds(N);
  for(int i=0;i<M;i++){
    int s,a,b;
    scanf("%d %d %d",&s,&a,&b);
    if(s==0)
      ds.merge(a,b);
    if(s==1){
      if(ds.find(a)==ds.find(b))
        printf("YES\n");
      else
        printf("NO\n");
    }
  }
}
```

factor_2.cpp

```
//아리스토테네스 체를 이용한 소인수분해
#include <cstdio>
#include <vector>

using namespace std;

int minFactor[10000001];

void eratosthenes(int n){
  minFactor[0] = minFactor[1] = -1;
  for(int i=2;i<=n;i++)
    minFactor[i]=i;
  for(int i=2;i*i<=n;i++){
    if(minFactor[i]==i){
      for(int j=i*i;j<=n;j+=i){
        if(minFactor[j]==j)
          minFactor[j]=i;
      }
    }
  }
}

int main(){
  int N;
  scanf("%d",&N);
  eratosthenes(N);
  vector<int> factor;
  while(N>1){
    factor.push_back(minFactor[N]);
    N/=minFactor[N];
  }
  for(int i=0;i<factor.size();i++){
    printf("%d\n",factor[i]);
```

```
  }
}
```

FenwickTree.cpp

```cpp
// 펜윅 트리 - 부분합을 빠르게 구함
#include <cstdio>
#include <vector>

using namespace std;

struct Fenwicktree {
  vector<int> tree;
  Fenwicktree(int n) : tree(n+1) {}
  int sum(int pos){
    ++pos;
    int ret = 0;
    while(pos > 0){
      ret += tree[pos];
      pos &= (pos-1);
    }
    return ret;
  }
  void add(int pos, int val){
    ++pos;
    while(pos < tree.size()){
      tree[pos] += val;
      pos += (pos & -pos);
    }
  }
};

int main(){
  int N,M,K;
  scanf("%d %d %d",&N,&M,&K);
  vector<int> v(N+1);
  Fenwicktree tree(N+1);
  for(int i=1;i<=N;i++){
    scanf("%d",&v[i]);
    tree.add(i,v[i]);
  }
  for(int i=0;i<M+K;i++){
    int A,B,C;
    scanf("%d %d %d",&A,&B,&C);
    if(A==1) tree.add(B,C-v[B]), v[B]=C;
    else printf("%d\n",tree.sum(C)-tree.sum(B-1));
  }
}
```

floyd.cpp

```cpp
// 플로이드-워셜 알고리즘
// mat[][] 전체를 무한대로 초기화 시키고 저장
// ans[][] 두 정점을 잇는 경로의 최소 비용 테이블
// ans[][]는 초기값이 mat[]와 같고 mat[i][i]=0으로!
// 3중포문 - (거치는 점 - 출발 점 - 도착점)순으로 한 후 비교
#include <cstdio>
#include <algorithm>
#define INF 987654321

using namespace std;

int N,M;
int mat[101][101],ans[101][101];

void floyd(){
  for(int i=1;i<=N;i++)
    for(int j=1;j<=N;j++)
      ans[i][j]=mat[i][j];
  for(int i=1;i<=N;i++)
    ans[i][i]=0;
  for(int k=1;k<=N;k++)
    for(int i=1;i<=N;i++)
      for(int j=1;j<=N;j++)
        if(ans[i][j]>ans[i][k]+ans[k][j])
          ans[i][j]=ans[i][k]+ans[k][j];
}
int main(){
  scanf("%d %d",&N,&M);
  for(int i=1;i<=N;i++)
    for(int j=1;j<=N;j++)
      mat[i][j]=INF;
  for(int i=0;i<M;i++){
    int A,B,C;
    scanf("%d %d %d",&A,&B,&C);
    mat[A][B]=min(mat[A][B],C);
  }
  floyd();
  for(int i=1;i<=N;i++){
    for(int j=1;j<=N;j++){
      printf("%d ",ans[i][j]);
    }
    printf("\n");
  }
}
```

KMP2.cpp

```cpp
//KMP 알고리즘
#include <cstdio>
#include <string>
#include <iostream>
#include <vector>

using namespace std;

//N에서 자기자신을 찾으면서 나타내는 부분일치를 이용해 pi계산
//pi[i]=N[..i]의 접미사도 되고 접두사도 되는 문자열의 최대 길이
vector<int> getPartialMatch(string& N){
  int n = N.size();
  vector<int> pi(n,0);
  int matched = 0;
  for(int i=1;i<n;i++){
    while(matched > 0 && N[i] != N[matched])
      matched = pi[matched-1];
    if(N[i] == N[matched]){
      matched++;
      pi[i] = matched;
    }
  }
  return pi;
}

//M의 부분문자열로 N이 출현하는 시작 위치들을 모두 반환
vector<int> kmpSearch(string& M, string& N){
  int m = M.size(), n = N.size();
  vector<int> ret;
  vector<int> pi = getPartialMatch(N);
  int matched = 0;
  for(int i=0;i<m;i++){
    while(matched > 0 && M[i] != N[matched])
      matched = pi[matched-1];
    if(M[i] == N[matched]){
      matched++;
      if(matched == n){
        ret.push_back(i-n+1);
        matched = pi[matched-1];
      }
    }
  }
  return ret;
}

int main(){
  string M,N;
  getline(cin,M);
  getline(cin,N);
  vector<int> ans = kmpSearch(M,N);
```

```
    printf("%lu\n",ans.size());
    for(int i=0;i<ans.size();i++){
      printf("%d\n",ans[i]);
    }
  }
```

kruskal.cpp

```cpp
// 크루스칼 알고리즘
#include <cstdio>
#include <vector>
#include <algorithm>

using namespace std;

int V,E;
const int MAX_V = 1001;
vector<pair<int, int> > adj[MAX_V];

struct DisjointSet{
  vector<int> parent, rank;
  DisjointSet(int n) : parent(n), rank(n,1){
    for(int i=0;i<n;i++)
      parent[i]=i;
  }
  int find(int u){
    if(u == parent[u]) return u;
    return parent[u] = find(parent[u]);
  }
  void merge(int u, int v){
    u = find(u); v = find(v);
    if(u == v) return;
    if(rank[u] > rank[v]) swap(u,v);
    parent[u] = v;
    if(rank[u] == rank[v]) ++rank[v];
  }
};

int kruskal(vector<pair<int,int> >& selected){
  int ret = 0;
  selected.clear();
  vector<pair<int,pair<int,int> > > edges;
  for(int u=0;u<V;u++){
    for(int i=0;i<adj[u].size();i++){
      int v = adj[u][i].first, cost = adj[u][i].second;
      edges.push_back(make_pair(cost, make_pair(u,v)));
    }
  }
  sort(edges.begin(),edges.end());
  DisjointSet sets(V);
  for(int i=0;i<edges.size();i++){
    int cost = edges[i].first;
    int u = edges[i].second.first, v= edges[i].second.second;
    if(sets.find(u)==sets.find(v)) continue;
    sets.merge(u,v);
    selected.push_back(make_pair(u,v));
```

```
    ret += cost;
  }
  return ret;
}

int main(){
  scanf("%d %d",&V,&E);
  for(int i=0;i<E;i++){
    int a,b,c;
    scanf("%d %d %d",&a,&b,&c);
    adj[a].push_back(make_pair(b,c));
  }
  vector<pair<int,int> > selected;
  printf("%d\n",kruskal(selected));
}
```

LCS.cpp

```
// LCS - 최장 공통 부분열 Longest Common Subsequence
// LCS(i,j) = LCS(i-1,j-1)+1 (s1[i]==s2[j])
//            max(LCS(i-1,j),LCS(i,j-1)) (s1[i]!=s2[j])
// LCS 뒤에서부터 보면서 출력
#include <cstdio>
#include <string>
#include <iostream>
#include <algorithm>

using namespace std;

int dp[1001][1001];
string s1,s2,ans;

string backTracking(int i, int j){
  if(i==0||j==0) return "";
  if(s1[i-1]==s2[j-1])
    return backTracking(i-1,j-1) + s1[i-1];
  else{
    if(dp[i][j-1] > dp[i-1][j])
      return backTracking(i,j-1);
    else
      return backTracking(i-1,j);
  }
}

int main(){
  cin >> s1 >> s2;
  for(int i=1;i<=s1.size();i++){
    for(int j=1;j<=s2.size();j++){
      if(s1[i-1]==s2[j-1]) dp[i][j]=dp[i-1][j-1]+1;
      else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
    }
  }
  printf("%d\n",dp[s1.size()][s2.size()]);

```

```
  int idx = dp[s1.size()][s2.size()];
  for(int i=s1.size();i>0;i--){
    for(int j=s2.size();j>0;j--){
      if(s1[i-1]==s2[j-1] && idx==dp[i][j]){
        ans=s1[i-1]+ans;
        idx--; i--;
      }
    }
  }
  printf("%s\n",ans.c_str()); // LCS출력 반복문 버전
  printf("%s\n",backTracking(s1.size(),s2.size()).c_str()); //LCS출력 재귀함수 버전
}
```

LIS.cpp

```
// LIS NLogN
// LIS의 마지막 값이 추가하는 배열의 값보다 작으면 추가.
// 그렇지 않으면 추가하는 배열의 값보다 큰 수 중 가장 작은 수와 교체.
// path[i] = LIS에서 i가 몇번 째에 있었는 지.
// 이후 뒤에서부터 보면서 출력.
#include <cstdio>
#include <vector>
#include <stack>

using namespace std;

vector<int> v;
int N, arr[1000001], path[1000001];

int main(){
  scanf("%d",&N);
  for(int i=0;i<N;i++){
    scanf("%d",&arr[i]);
    auto it = lower_bound(v.begin(),v.end(),arr[i]);
    path[i] = (int)(it-v.begin())+1;
    if(it==v.end()) v.push_back(arr[i]);
    else *it = arr[i];
  }
  printf("%lu\n",v.size());

  //경로 탐색
  stack<int> s;
  int idx = N-1;
  for(int i=v.size();i>0;idx--){
    if(path[idx]==i){
      s.push(arr[idx]); i--;
    }
  }
  while(!s.empty()){
    printf("%d ",s.top());
    s.pop();
  }
```

```
    printf("\n");
}
```

muti1.cpp

```
//큰 두수를 곱하는 n^2 알고리즘
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;
void normalize(vector<int>& num){
    num.push_back(0);
    for(int i=0;i<num.size();i++){
        num[i+1]+=num[i]/10;
        num[i]%=10;
    }
    while(num.size()>1&&num.back()==0) num.pop_back();
}

vector<int> multiply(const vector<int>& a, const vector<int>& b){
    vector<int> c(a.size()+b.size()+1,0);
    for(int i=0;i<a.size();i++)
        for(int j=0;j<b.size();j++)
            c[i+j]+=a[i]*b[j];
    normalize(c);
    return c;
}

int main(){
    char x[100]={0},y[100]={0};
    int xlen,ylen;
    scanf("%s %s",x,y);
    xlen=strlen(x); ylen=strlen(y);
    vector<int> a(xlen);
    vector<int> b(ylen);
    for(int i=0;i<xlen;i++) a[i]=x[xlen-1-i]-'0';
    for(int i=0;i<ylen;i++) b[i]=y[ylen-1-i]-'0';
    vector<int> c = multiply(a,b);
    for(int i=c.size()-1;i>=0;i--)
        printf("%d",c[i]);
    printf("\n");
}
```

newworkFlow.cpp

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <vector>

using namespace std;

```

```cpp
const int MAX_V = 1001;
const int INF = 987654321;
int V;
//capacity[u][v] = u에서 v로 보낼 수 있는 용량
//flow[u][v] = u에서 v로 흘러가는 유량 (방향이 반대일 경우 음수)
int capacity[MAX_V][MAX_V], flow[MAX_V][MAX_V];
//flow[][]를 계산하고 총 유량을 반환
int networkFlow(int source, int sink){
  memset(flow,0,sizeof(flow));
  int totalFlow = 0;
  while(true){
    vector<int> parent(MAX_V,-1);
    queue<int> q;
    parent[source] = source;
    q.push(source);
    while(!q.empty() && parent[sink] == -1){
      int here = q.front(); q.pop();
      for(int there = 0; there < V; there++){
        if(capacity[here][there] - flow[here][there] > 0 &&
          parent[there] == -1) {
          q.push(there);
          parent[there] = here;
        }
      }
    }
    if(parent[sink] == -1) break;
    int amount = INF;
    for(int p = sink; p != source; p = parent[p])
      amount = min(capacity[parent[p]][p] - flow[parent[p]][p],amount);
    for(int p = sink; p != source; p = parent[p]){
      flow[parent[p]][p] += amount;
      flow[p][parent[p]] -= amount;
    }
    totalFlow += amount;
  }
  return totalFlow;
}
```

binarySearch.cpp

```cpp
// 이분 탐색 틀
#include <cstdio>

int N,M,arr[10001],large;

bool decision(int cost){
  int ret=0;
  for(int i=0;i<N;i++)
    ret += arr[i]>cost ? cost : arr[i];
  return ret<=M;
}

```

```cpp
int main(){
  scanf("%d",&N);
  for(int i=0;i<N;i++){
    scanf("%d",&arr[i]);
    if(arr[i]>large) large=arr[i];
  }
  scanf("%d",&M);
  int lo=-1, hi=large+1;
  while(lo+1<hi){
    int mid=(lo+hi)/2;
    if(decision(mid))
      lo=mid;
    else
      hi=mid;
  }
  printf("%d\n",lo);
}
```

Nqueen.cpp

```cpp
//N-Queen 백트래킹
#include <cstdio>
#include <cstdlib>

int N,mat[15],ans;

bool isPromising(int y){
  for(int i=0;i<y;i++){
    if(mat[i]==mat[y] || abs(i-y)==abs(mat[i]-mat[y]))
      return false;
  }
  return true;
}

void queens(int y){
  if(y==N){
    ans++; return;
  }
  for(int i=0;i<N;i++){
    mat[y]=i;
    if(isPromising(y)) queens(y+1);
  }
}

int main(){
  scanf("%d",&N);
  queens(0);
  printf("%d\n",ans);
}
```

RMQ.cpp

```cpp
//RMQ 구간 최소 쿼리
#include <cstdio>
#include <vector>

using namespace std;

const int MAX_INT = numeric_limits<int>::max();

struct RMQ{
  int n;
  //각 구간의 최소치
  vector<int> rangeMin;
  RMQ(const vector<int>& array){
    n = array.size();
    rangeMin.resize(n * 4);
    init(array, 0, n-1, 1);
  }
  //node노드가 array[left..right]배열을 표현할 때
  //node를 루트로 하는 서브트리를 초기화하고 이 구간의 최소치를 반환
  int init(const vector<int>& array, int left, int right, int node){
    if(left == right) return rangeMin[node] = array[left];
    int mid = (left + right) / 2;
    int leftMin = init(array, left, mid, node*2);
    int rightMin = init(array, mid+1, right, node*2+1);
    return rangeMin[node] = min(leftMin, rightMin);
  }
  //node가 표현하는 범위 array[nodeLeft..nodeRight]가 주어질 때
  //이 범위와 array[left..right]의 교집합의 최소치를 구한다.
  int query(int left, int right, int node, int nodeLeft, int nodeRight){
    if(right < nodeLeft || nodeRight < left) return MAX_INT;
    if(left <= nodeLeft && nodeRight <= right) return rangeMin[node];
    int mid = (nodeLeft + nodeRight) / 2;
    return min(query(left, right, node*2, nodeLeft, mid),
               query(left, right, node*2+1, mid+1, nodeRight));
  }
  int query(int left, int right){
    return query(left, right, 1, 0, n-1);
  }
  //array[index]=newValue로 바뀌었을 때 node를 루트로 하는 구간트리를 갱신
  int update(int index, int newValue, int node, int nodeLeft, int nodeRight){
    if(index < nodeLeft || nodeRight < index) return rangeMin[node];
    if(nodeLeft == nodeRight) return rangeMin[node] = newValue;
    int mid = (nodeLeft + nodeRight) / 2;
    return rangeMin[node] = min(
      query(index, newValue, node*2, nodeLeft, mid),
      query(index, newValue,node*2+1, mid+1, nodeRight));
  }
  int update(int index, int newValue){
    return update(index, newValue, 1, 0, n-1);
  }
};
```

SA_LCP.cpp

```cpp
// s의 접미사 배열과 LCP(Longest Common Prefix).
// SA: O(N*logN*logN)
// SA: 정렬의 NlogN * 1,2,4,8로 보는 과정의 logN
// LCP: O(N)
#include <cstdio>
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>

using namespace std;

struct Comparator{
  const vector<int>& group;
  int t;
  Comparator(const vector<int>& _group, int _t): group(_group), t(_t){}
  bool operator() (int a, int b){
    // 첫 t글자가 다르면 이들을 이용해 비교.
    if(group[a] != group[b])
      return group[a] < group[b];
    // S[a+t..]와 S[b+t..]의 첫 t글자를 비교.
    return group[a+t] < group[b+t];
  }
};

vector<int> getSuffixArray(const string& s){
  int n = s.size();
  int t = 1;
  // group[i] = 접미사들을 첫 t글자 기준으로 정렬했을 때, S[i..]가 들어가는 그룹 번호
  // t=1일 때에는 정렬할 것 없이 S[i..]의 첫 글자로 그룹 번호를 정해 줘도 같은 효과
  vector<int> group(n+1);
  for(int i=0;i<n;i++)
    group[i] = s[i];
  group[n] = -1;
  // 결과적으로 접미사 배열이 될 반환 값. 이 배열을 logN번 정렬
  vector<int> perm(n);
  for(int i=0;i<n;i++)
    perm[i] = i;
  while(t < n){
    Comparator compareUsing2T(group,t);
    sort(perm.begin(),perm.end(),compareUsing2T);
    t*=2;
    if(t>=n) break;
    // 2t글자를 기준으로 한 그룹 정보.
    vector<int> newGroup(n+1);
    newGroup[n] = -1;
    newGroup[perm[0]] = 0;
```

```
48      for(int i=1;i<n;i++)
49        if(compareUsing2T(perm[i-1],perm[i]))
50          newGroup[perm[i]] = newGroup[perm[i-1]] + 1;
51        else
52          newGroup[perm[i]] = newGroup[perm[i-1]];
53        group = newGroup;
54    }
55    return perm;
56  }
57
58  // 접미사 배열내의 인접한 접미사와 비교.
59  // LCP[i] = SA[i]와 SA[i-1]를 인덱스로 가지는 S[i..] 와 S[j..]의 가장 긴 공통 접두사
60  // 전 len정보를 가지고 len 업데이트 이후 LCP[i]에 대입.
61  vector<int> getLCP(const string& s, vector<int> SA){
62    int n = s.size();
63    vector<int> LCP(n);
64    vector<int> rank(n);
65    for(int i=0;i<n;i++)
66      rank[SA[i]]=i;
67    int len = 0;
68    for(int i=0;i<n;i++){
69      if(rank[i]){
70        int j = SA[rank[i]-1];
71        while(s[j+len] == s[i+len]) len++;
72        LCP[rank[i]] = len;
73      }
74      len = max(len-1,0);
75    }
76    return LCP;
77  }
78
79  int main(){
80    string s;
81    cin >> s;
82    vector<int> SA = getSuffixArray(s);
83    vector<int> LCP = getLCP(s,SA);
84    for(int i=0;i<SA.size();i++)
85      printf("%d ",SA[i]);
86    printf("\nx ");
87    for(int i=1;i<LCP.size();i++)
88      printf("%d ",LCP[i]);
89    printf("\n");
90  }
```

segment.cpp

```cpp
//세그먼트트리
//리프노드: 배열의 그 수 자체
//다른노드: 왼쪽 자식과 오른쪽 자식의 합을 저장
#include <stdio.h>
#include <math.h>
#include <vector>

using namespace std;

long long init(vector<long long> &a, vector<long long> &tree, int node, int start, int end){ //초기 트리 생성
  if (start==end) return tree[node]=a[start];
  else return
  tree[node]=init(a,tree,node*2,start,(start+end)/2)+init(a,tree,node*2+1,(start+end)/2+1,end);
  //왼쪽 자식과 오른쪽 자식의 합
}
void update(vector<long long> &tree, int node, int start, int end, int index, long long diff){
  //여기서 diff=val-a[index];
  if (index<start || index>end) return;
  tree[node] = tree[node]+diff;
  if (start != end){
    update(tree,node*2,start,(start+end)/2,index,diff);
    update(tree,node*2+1, (start+end)/2+1, end, index, diff);
  }
}
// node가 담당하는 구간이 start~end이고, 구해야하는 합의 범위는 left~right
long long sum(vector<long long> &tree, int node, int start, int end, int left, int right){
  if (left>end || right<start) return 0;
  if (left<=start && end<=right) return tree[node];
  return sum(tree,node*2,start,(start+end)/2,left,right) +
  sum(tree,node*2+1,(start+end)/2+1,end,left,right);
}

int main(){
  int N,M,K; //수의 개수 N, 수의 변경 M, 구간의 합 K
  scanf("%d %d %d",&N,&M,&K);
  vector<long long> a(N);
  int H = (int)ceil(log2(N));
  int tree_size = (1<<(H+1)); //필요한 배열의 크기 = 2^(H+1)-1;
  vector<long long> tree(tree_size);
  for(int i=0;i<N;i++) scanf("%lld",&a[i]);
  init(a,tree,1,0,N-1); //tree[1]에 a[0]~a[N-1]까지의 합
  int all = M+K;
  while(all--){
    int A;
    scanf("%d",&A);
    if(A==1){ //수의 변경 b번째 수를 c로 바꾼다
      int B; long long C;
      scanf("%d %lld",&B,&C);
      B--;
      long long diff = C-a[B]; a[B]=C;
      update(tree,1,0,N-1,B,diff);
    }
```

```
    else if(A==2){ //배열의 합 b부터 c까지의 합을 출력
      int B,C;
      scanf("%d %d",&B,&C);
      printf("%lld\n",sum(tree,1,0,N-1,B-1,C-1));
    }
  }
  return 0;
}
```

str_perm.cpp

```
// 문자열 조합 개수
// 알고리즘 원리 - 같은 숫자가 여러 개 남아있다면 그 중 가장 먼저 오는 것만 선택
// 1. 없거나 2. 이번 자리수랑 다르거나 3. 이미 사용되었을 경우에만 선택
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <string>

using namespace std;

string e,digits;
int n;
bool taken[15];

void generate(string price, bool taken[15]){
  if(price.size()==n){
    cout << price << endl;
    return;
  }
  for(int i=0;i<n;i++){
    if(!taken[i] && (i==0 || digits[i-1] != digits[i] || taken[i-1])){
      taken[i] = true;
      generate(price + digits[i], taken);
      taken[i] = false;
    }
  }
}

int main(){
  cin >> e;
  n = e.size();
  digits = e;
  sort(digits.begin(),digits.end());
  generate("",taken);
}
```

twopoint.cpp

```cpp
// 가장 가까운 두 점
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <algorithm>
#define INF 987654321
using namespace std;

vector<pair<int, int> > v;

int dist(pair<int, int>& p1, pair<int, int>& p2){
  return
  (p1.first-p2.first)*(p1.first-p2.first)+(p1.second-p2.second)*(p1.second-p2.second);
}

int foo(int left, int right){
  if(left >= right) return INF;
  if(right - left == 1) return dist(v[left],v[right]);
  int mid = (left+right)/2;
  int ret = min(foo(left,mid),foo(mid+1,right));
  for(int i=mid; i>=left; i--){
    int dx = v[mid].first-v[i].first;
    if(dx*dx >= ret) break;
    for(int j=mid+1;j<=right;j++){
      int dx = v[j].first-v[i].first;
      if(dx*dx >= ret) break;
      ret = min(ret,dist(v[i],v[j]));
    }
  }
  return ret;
}

int main(){
  int N;
  scanf("%d",&N);
  v.resize(N);
  for(int i=0;i<N;i++)
    scanf("%d %d",&v[i].first,&v[i].second);
  sort(v.begin(),v.end());
  printf("%d\n",foo(0,N-1));
}
```

vector.cpp

```cpp
//다각형의 면적
#include <cstdio>
#include <cmath>
#include <vector>
#include <algorithm>

using namespace std;

struct vector2{
  double x,y;
  explicit vector2(double x_ = 0, double y_ = 0) :x(x_), y(y_){}
};

double area(const vector<vector2>& p){
  double ret = 0;
  for(int i=0;i<p.size();i++){
    int j=(i+1)%p.size();
    ret += p[i].x * p[j].y - p[j].x * p[i].y;
  }
  return fabs(ret) / 2.0;
}

int main(){
  int N;
  scanf("%d",&N);
  vector<vector2> v;
  for(int i=0;i<N;i++){
    int X,Y;
    scanf("%d %d",&X,&Y);
    v.push_back(vector2(X,Y));
  }
  printf("%.1f\n",area(v));
}
```

```cpp
// 볼록껍질
#include <cstdio>
#include <cmath>
#include <vector>
#include <algorithm>

using namespace std;

const double PI = 2.0 * acos(0.0);

struct vector2{
  double x,y;
  explicit vector2(double x_ = 0, double y_ = 0) :x(x_), y(y_){}
  bool operator == (const vector2& rhs) const{
    return x == rhs.x && y == rhs.y;
  }
  bool operator < (const vector2& rhs) const{
```

```
    return x != rhs.x ? x < rhs.x : y < rhs.y;
  }
  vector2 operator + (const vector2& rhs) const{
    return vector2(x + rhs.x, y + rhs.y);
  }
  vector2 operator - (const vector2& rhs) const{
    return vector2(x - rhs.x, y - rhs.y);
  }
  double norm() const{
    return hypot(x,y);
  }
  vector2 normalize() const{
    return vector2(x/norm(),y/norm());
  }
  double dot(const vector2& rhs) const{
    return x * rhs.x + y * rhs.y;
  }
  double cross(const vector2& rhs) const{
    return x * rhs.y - rhs.x * y;
  }
};

double ccw(vector2 a, vector2 b){
  return a.cross(b);
}
double ccw(vector2 p, vector2 a, vector2 b){
  return ccw(a-p,b-p);
}

typedef vector<vector2> polygon;
int giftWrap(const vector<vector2>& points){
  int n = points.size(), ret=0;
  polygon hull;
  vector2 pivot = *min_element(points.begin(),points.end());
  hull.push_back(pivot); ret++;
  while(true){
    vector2 ph = hull.back(), next = points[0];
    for(int i=1;i<n;i++){
      double cross = ccw(ph,next,points[i]);
      double dist = (next-ph).norm()-(points[i]-ph).norm();
      if(cross>0 || (cross==0 && dist<0))
        next=points[i];
    }
    if(next==pivot) break;
    hull.push_back(next); ret++;
  }
  return ret;
}
```

```
66
67  int main(){
68    int N;
69    scanf("%d",&N);
70    vector<vector2> v;
71    for(int i=0;i<N;i++){
72      int X,Y;
73      scanf("%d %d",&X,&Y);
74      v.push_back(vector2(X,Y));
75    }
76    printf("%d\n",giftWrap(v));
77  }
```

INDEX