STL

- Min Stack

```
class minStack{
	public:
	stack<int> pilha;
	int min = INT_MAX;

	void push(int x){
		if (pilha.empty()){
			pilha.push(x);
			min = x;
		}else{
			if (x >= min) pilha.push(x);
			else{
				pilha.push(2*x-min);
				min = x;
			}
		}
	}
	int pop(){
		if (pilha.empty()){
			return -1;
		}
		int y = pilha.top();
		pilha.pop();
		if (y < min){
			min = 2*min - y;
		}
		return 1;
	}
	int getMin(){
		if (pilha.empty()) return -1;
		return this->min;
	}
};
```

BACKTRACK

- Sudoku

```
vector<vi > sudoku;
int n, aux;
bool done = false;

bool canPlace(int k, int x, int y){
   //check row
```

```
    for (int i = x, j = 0; j < n*n; j++)
        if (sudoku[i][j] == k) return false;
    //check column
    for (int i = 0, j = y; i < n*n; i++)
        if (sudoku[i][j] == k) return false;
    //check cell
    int rowI, rowE, colI, colE;
    if (n == 3){
        if (x <= 2) {rowI = 0; rowE = 2;}
        else if (3 <= x && x <= 5) {rowI = 3; rowE = 5;}
        else {rowI = 6; rowE = 8;}

        if (y <= 2) {colI = 0; colE = 2;}
        else if (3 <= y && y <= 5) {colI = 3; colE = 5;}
        else {colI = 6; colE = 8;}
    }else{
        if (x <= 1) {rowI = 0; rowE = 1;}
        else {rowI = 2; rowE = 3;}

        if (y <= 1) {colI = 0; colE = 1;}
        else {colI = 2; colE = 3;}
    }
    for (int i = rowI; i <= rowE; i++)
        for (int j = colI; j <= colE; j++)
            if (sudoku[i][j] == k) return false;

    return true;
}

void printSudoku(){
    for (int i = 0; i < n*n;i++){
        cout << sudoku[i][0];
        for (int j = 1; j < n*n; j++){
            cout << " " << sudoku[i][j];
        }
        cout << endl;
    }
}

void backTrack(int i, int j){
    if (done) return;
    if (i > (n*n)-1){
        printSudoku();
        done = true;
        return;
    }
    // printSudoku(); cout << endl;
    if (sudoku[i][j]){
        if (j+1 < n*n) backTrack(i, j+1);
        else backTrack(i+1, 0);
        return;
    }

    // for (int k = n*n; k >= 1; k--){
    for (int k = 1; k <= n*n; k++){
        if (canPlace(k, i, j)){
            sudoku[i][j] = k;
            if (j+1 < n*n) backTrack(i, j+1);
            else backTrack(i+1, 0);
            sudoku[i][j] = 0;
        }
    }
}

int main(){
    ios::sync_with_stdio(false), cin.tie(0);
    bool first = true;
    while(cin >> n){
        if (!first) cout << endl;
        first = false;
        sudoku.resize(0);
        done = false;
        sudoku.resize(n*n);
        for (int i = 0; i < n*n;i++){
            sudoku[i].resize(n*n);
            for (int j = 0; j < n*n; j++){
                cin >> aux; sudoku[i][j] = (aux);
            }
        }
        if(n>1){
            backTrack(0, 0);
            if(!done) cout << "NO SOLUTION\n";
        }
        else cout << "1\n";
    }
    return 0;
```

```
}
```

- Sum it up

```
int S, n;
vi numbers;
vb uses;
set<string> printing;

void printUses(){
	string output = "";
	for (int i = 0; i < uses.size(); i++){
		if (uses[i])
			output += to_string(numbers[i]) + "+";
	}
	output = output.substr(0, output.size()-1) + "\n";
	printing.insert(output);
}

void backTrack(int i, int sum, int rem){
	if (sum == S){
		printUses();
		return;
	}

	if (i == numbers.size() || sum >= S || rem == 0) return;

	if (sum + rem >= S && sum+numbers[i] <= S){
		uses[i] = true;
		backTrack(i+1, sum + numbers[i], rem - numbers[i]);
	}

		uses[i] = false;
	if (sum >= S) return;
		backTrack(i+1, sum, rem - numbers[i]);
}

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

	while(true){
		cin >> S >> n;
		if (S + n == 0) return 0;
		int sum = 0, aux;
		numbers.resize(0);
		uses.resize(0);
		printing.clear();
		while(n--){
			cin >> aux;
			sum += aux;
			numbers.push_back(aux);
			uses.push_back(false);
		}
		cout << "Sums of " << S << ":\n";
		backTrack(0, 0, sum);
		string outt = "";
		for (auto i: printing){
			outt = i + outt;
		}
		if (outt == "") cout << "NONE\n";
		cout << outt;

	}
	return 0;
}
```

- Password

```
vs grid1, grid2;
set<string> pass;

bool possible(char x, int col){
	for (int i = 0; i < 6; i++){
		if (grid2[i][col] == x){
			return true;
		}
	}
	return false;
}

void backTrack(int k, string password){
	if (k == 5){
		pass.insert(password);
		return;
	}
	for (int i = 0; i < 6; i++){
		if (possible(grid1[i][k], k))
			backTrack(k+1, password+grid1[i][k]);
	}
}

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

    int casos, K; cin >> casos;

    while(casos--){
        cin >> K;
        grid1.resize(0);
        grid2.resize(0);
        pass.clear();
        string aux;
        for (int i = 0; i < 6; i++){
            cin >> aux;
            grid1.push_back(aux);
        }
        for (int i = 0; i < 6; i++){
            cin >> aux;
            grid2.push_back(aux);
        }
        backTrack(0, "");
        if (K > pass.size())
            cout << "NO\n";
        else{
            int contador = 0;
            for(auto x: pass){
                contador++;
                if (contador == K){
                    cout << x << endl;
                    break;
                }
            }
        }
    }
    return 0;
}
```

DFS BFS

- Two Buttons

```
ll n, m;
vi vis;
int BFS(){
        queue<ii > q;
        q.push({n, 0});
        vis[n] = 1;

        while (!q.empty()){
                ii u = q.front(); q.pop();
                vis[u.x] = 1;
                if (u.x == m) return u.y;
                if (u.x*2 <= 10000 && vis[u.x*2] == 0) q.push({u.x*2, u.y + 1});
                if (u.x-1 > 0 && vis[u.x-1] == 0) q.push({u.x-1, u.y + 1});
        }
        return -1;
}

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

        vis.resize(10001, 0);
        cin >> n >> m;
        cout << BFS() << endl;

        return 0;
}
```

- Lexicographically smallest sequence of nodes

```
int nodes, edges;
vector<si > adj;
vi vis; vi out;
string output = "";

void BFS(int u){
        priority_queue<int, vector<int>, greater<int>> q;
        q.push(u);

        while(!q.empty()){
                if (out.size() == nodes) return;
                int v = q.top(); q.pop();
                if (vis[v]) continue;
                if (!vis[v]) {out.push_back(v); output += to_string(v) + " ";}
                vis[v] = 1;

                for (auto x : adj[v]){
                        if(!vis[x]) q.push(x);
                }
        }
}

int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        cin >> nodes >> edges;
        vis.resize(nodes+2, 0);
        adj.resize(nodes+2, si());

        for (int i = 0; i < edges; i++){
                int v1, v2; cin >> v1 >> v2;
                adj[v1].insert(v2);
                adj[v2].insert(v1);
        }

        BFS(1);

        cout << output.substr(0, output.size()-1) << endl;

        return 0;
}
```

DSU e Caminhos mínimos

- Igor in the museum (Ver quantas paredes tem)

```
int n, m, k;
int matrix[1005][1005], vis[1000005];
map<ii, int> indice;
int ds[1000005], ans[1000005];
void dsBuild(){
   for (int i = 0; i < n*m+1; i ++){
      ds[i] = i;
      ans[i] = 0;
   }
}
int dsFind(int i){
   if (ds[i] != i) ds[i] = dsFind(ds[i]);
   return ds[i];
}
void dsUnion(int a, int b){
   a = dsFind(a); b = dsFind(b);
   int A = ans[a]; ans[a] = 0;
```

```
    int B = ans[b]; ans[b] = 0;
    ds[b] = a;
    ans[dsFind(a)] = A+B;
}

int main(){
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n >> m >> k;
    int index = 0;
    dsBuild();
    for (int i = 0; i < n; i++){
        string aux; cin >> aux;
        for (int j = 0; j < m; j++){
            indice[{i,j}] = index++;
            (aux[j] == '.') ? matrix[i][j] = 1 : matrix[i][j] = 0;
            if (i-1 >= 0){
                if (matrix[i][j]){
                    if (matrix[i-1][j] == 0) ans[dsFind(indice[{i,j}])]++;
                    else dsUnion(indice[{i,j}], indice[{i-1,j}]);
                }else{
                    if (matrix[i-1][j] == 1) ans[dsFind(indice[{i-1,j}])]++;
                }
            }
            if (j-1 >= 0){
                if (matrix[i][j]){
                    if (matrix[i][j-1] == 0) ans[dsFind(indice[{i,j}])]++;
                    else dsUnion(indice[{i,j}], indice[{i,j-1}]);
                }else{
                    if (matrix[i][j-1] == 1)
                        ans[dsFind(indice[{i,j-1}])]++;
                }
            }
        }
    }
    while(k--){
        int u,v; cin >> u >> v; u--;v--;
        cout << ans[dsFind(indice[{u,v}])] << endl;
    }
        return 0;
}
```

DP

- Longest Palindrome

```
string str1, str2;
int dp[1005][1005];
int LCS(int i, int j){
        if (dp[i][j] != -1)
                return dp[i][j];
        if (str1[i] == '#' || str2[j] == '#')
                return dp[i][j] = 0;
        if (str1[i] == str2[j])
                return dp[i][j] = 1 + LCS(i+1, j+1);

        return dp[i][j] = max(LCS(i+1,j), LCS(i,j+1));
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        int Q; cin >> Q;
        cin.ignore();
        while(Q--){
                memset(dp, -1, sizeof(dp));
                getline(cin, str1); str2 = "";
                for (auto x : str1)
                        str2 = x + str2;
                str1 += "#"; str2 += "#";
                //cout << str1 << " " << str2 << endl;
                cout << LCS(0, 0) << endl;
        }
        return 0;
}
```

- Palindrome 2000

```
string str1, str2;
int L[maxn][maxn];
int lcs(int m, int n){
        int i, j;
        for (i = 0; i <= m; i++){
                for (j = 0; j <= n; j++){
                        if (i==0 || j== 0)
                                L[i][j] = 0;
                        else if (str1[i-1] == str2[j-1])
                                L[i][j] = L[i-1][j-1]+1;
                        else
```

```
                    L[i][j] = max(L[i-1][j], L[i][j-1]);
            }
        }
        return L[m][n];
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        int n; cin >> n;
        cin >> str1; str2 = "";
        for (auto x : str1) str2 = x + str2;
        cout << n-lcs(n, n) << endl;
        return 0;
}
```

- Take the land

```
int N, M;
int mat[maxn][maxn], aux[maxn][maxn];
void preProcess(){
        for (int i = 0; i < M; i++)
                aux[0][i] = mat[0][i];

        for (int i = 1; i < N; i++)
                for (int j = 0; j < M; j++)
                        aux[i][j] = mat[i][j] + aux[i-1][j];

        for (int i = 0; i < N; i++)
                for (int j = 1; j < M; j++)
                        aux[i][j] += aux[i][j-1];
}
int sumQuery(int tli, int tlj, int rbi, int rbj){
        int res = aux[rbi][rbj];
        if (tli>0)
                res -= aux[tli-1][rbj];
        if (tlj>0)
                res -= aux[rbi][tlj-1];
        if (tli>0 && tlj>0)
                res += aux[tli-1][tlj-1];
        return res;
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        while (cin >> N >> M && N){
```

```
		for (int i = 0; i < N; i++)
			for (int j = 0; j < M; j++)
				cin >> mat[i][j];
		preProcess();
		int size = 0;
		for (int tli = 0; tli < N; tli++)
		 for (int tlj = 0; tlj < M; tlj++)
		  for (int rbi = tli; rbi < N; rbi++)
			for (int rbj = tlj; rbj < M; rbj++){
				int sum = sumQuery(tli,tlj,rbi,rbj);
				if (sum > 0) continue;
				size = max(size, (rbi-tli+1)*(rbj-tlj+1));
			}
		cout << size << endl;
	}
	return 0;
}
```

- DIE HARD

```
int dp[1005][1005];
/*
air = 0
water = 1
fire = 2
*/
int calc(int h, int a, int s){
	if (dp[h][a] != 0) return dp[h][a];
	//printf("%d %d %d\n", s, h, a);
	int best = 0;
	if (s == 0){
		int water = -1, fire = -1;
		if (h-20 > 0 && a+5 > 0) fire = calc(h-20, a+5, 2);//f
		if (h-5 > 0 && a-10 > 0) water = calc(h-5, a-10, 1);//w
		best =max(water, fire) + 1;
	}else{
		best = calc(h+3, a+2, 0) + 1;
	}
	return dp[h][a] = max(best, dp[h][a]);
}
int main(){
	ios::sync_with_stdio(false), cin.tie(0);
	int T; cin >> T;
	int H, A;
	while(T--){
		memset(dp, 0, sizeof(dp));
		cin >> H >> A;
		cout << calc(H+3, A+2, 0)+1 << endl;
	}

	return 0;
}
```

- Happy VALENTINE (TSP)

```
int m, n;
int graph[ms][ms], dp[ms][1<<15], dist[ms*ms], dist_destiny[ms*ms];
vector<vi > adj(ms*ms, vi());
vs mat; ii robot, deliver;
vector<ii > pts;
void bfs(int u){
	memset(dist, -1, sizeof(dist));
	queue<int> q; q.push(u);
	dist[u] = 0;

	while(!q.empty()){
		int u = q.front(); q.pop();
		for (auto v : adj[u]){
			if (dist[v] == -1){
				dist[v] = dist[u]+1;
				q.push(v);
			}
		}
	}
}
//TSP
int N;
int solve(int pos, int visited){
	N = pts.size();
	if (visited+1 == (1<<(N+1))){
		if (!N)
			return dist_destiny[robot.a*m + robot.b];
		if (pos<1) return 9999;
		return dist_destiny[pts[pos-1].a*m + pts[pos-1].b];
	}
```

```
        int &ans = dp[pos][visited];
        if (~ans) return ans;

        ans = inf;

        for (int i = 0; i < N+1; i++){
                if ( !(visited & (1<<i)) )
                        ans = min(ans, graph[pos][i] + solve(i,visited | (1<<i)));
        }
        return dp[pos][visited] = ans;
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        int Q; cin >> Q;
        while(Q--){
                //--Reset--//
                cin >> n >> m;
                memset(graph, 0, sizeof(graph));
                memset(dp, -1, sizeof(dp));
                memset(dist, -1, sizeof(dist));
                mat.resize(0);
                adj.resize(0); adj.resize(ms*ms, vi());
                pts.resize(0);
                robot = {-1,-1};
                deliver = {-1,-1};
                //---------//
                for (int i = 0; i < n; i++){
                        string line; cin >> line; mat.pb(line);
                        for (int j = 0; j < m; j++){
                                if (line[j] == 'T')
                                        robot = {i,j};
                                else if (line[j] == 'W')
                                        deliver = {i,j};
                                else if (line[j] == 'C')
                                        pts.push_back({i,j});

                                if (line[j] == '#') continue;
                                //* BFS part
                                if (i > 0 && mat[i-1][j] != '#'){
                                        adj[i*m+j].pb((i-1)*m+j);
                                        adj[(i-1)*m+j].pb(i*m+j);
                                }
                                if (j > 0 && mat[i][j-1] != '#'){
                                        adj[i*m+j].pb(i*m+j-1);
                                        adj[i*m+j-1].pb(i*m+j);
                                }//*/
                        }
                }
                bfs(robot.a*m + robot.b);
                bool possible = true;
                for (auto x : pts){
                        if (dist[x.a*m+x.b] == -1)
                                possible = false;
                }
                if (!possible){
                        cout << "Mission Failed!" << endl;
                        if(Q) cout << endl;
                        continue;
                }
                //first is bfs for robot wich is done
                graph[0][0] = 0;
                for (int j = 1; j <= int(pts.size()); j++){
                        graph[0][j] = graph[j][0] = dist[pts[j-1].a*m + pts[j-1].b];
                }
                //bfs from all points and robot to the deliver point
                bfs(deliver.a*m + deliver.b);
                for (int i = 0; i < ms*ms; i++)
                        dist_destiny[i] = dist[i];

                if (dist[robot.a*m + robot.b] == -1){
                        cout << "Mission Failed!" << endl;
                        if(Q) cout << endl;
                        continue;
                }
                for (int i = 1; i <= int(pts.size()); i++){
                        bfs(pts[i-1].a*m + pts[i-1].b);

                        for (int j = 1; j <= int(pts.size()); j++){
                                graph[i][j] = dist[pts[j-1].a*m + pts[j-1].b];
                        }
                }
                /* A matriz q o TSP usa, a dist entre p a p e robo a ponto
```

```
			for (int i = 0; i <= int(pts.size()); i++){
				for (int j = 0; j <= int(pts.size()); j++){
					cout << graph[i][j] << " ";
				}
				cout << endl;
			}//*/
			cout << solve(0,0) << endl;
			if(Q) cout << endl;
		}
		return 0;
}
```

- 369 Numbers (Digit DP)

```
int dp[50][2][2][18][18][18];
vi numberA, numberB;
int solve(int pos, int smaller, int bigger, int t, int s, int n){
		if (t > 17 || s > 17 || n > 17) return 0;

		if (pos == int(numberA.size()))
			return (t > 0 && t == s && t == n);
		if (smaller && bigger){
			if (dp[pos][smaller][bigger][t][s][n] != -1) return
dp[pos][smaller][bigger][t][s][n];
		}
		ll ans = 0;
		int limSup = (smaller) ? 9 : numberB[pos];
		int limInf = (bigger) ? 0 : numberA[pos];

		for (int digit = limInf; digit <= limSup; digit++){
			int new_smaller = smaller, new_bigger = bigger;

			if (!smaller && digit < limSup)  new_smaller = 1;
			if (!bigger && digit > limInf)  new_bigger = 1;

			ans += solve(pos+1, new_smaller, new_bigger, t+(digit == 3), s+(digit
== 6), n+(digit == 9));
			ans %= inf;
		}

		return dp[pos][smaller][bigger][t][s][n] = ans;
}
```

```
int main(){
        FAST;
        int t; cin >> t;
                memset(dp, -1, sizeof(dp));
        while(t--){
            string a, b; cin >> a >> b;
            //Solve
                numberA.resize(0); numberB.resize(0);
                if (a.size() < 50){
                    int times = 50-a.size();
                    while(times--)
                        numberA.push_back(0);
                }
                if (b.size() < 50){
                    int times = 50-b.size();
                    while(times--)
                        numberB.push_back(0);
                }

                for (auto x : a)
                    numberA.push_back(x-'0');
                for (auto x : b)
                    numberB.push_back(x-'0');

            printf("%d\n", solve(0,0,0,0,0,0));
            if (t!=0) printf("\n");
        }

        return 0;
}
```

- Count the indexes

```
vector<vi> mapa; vi num;
int lookFor(int numb, int i, int j){
        int loww = lower_bound(mapa[numb].begin(), mapa[numb].end(), i) -
mapa[numb].begin();
        int highh = upper_bound(mapa[numb].begin(), mapa[numb].end(), j) -
mapa[numb].begin();

        return highh-loww;
}
int main(){
	ios::sync_with_stdio(false), cin.tie(0);

	int t; cin >> t;
	mapa.resize(200005, vi());
	while(t--){
		int com; cin >> com;

		if (com == 0){
			if (num.size() > 0){
				int last = num.back();
				num.pop_back();
				mapa[last].pop_back();
			}else cout << "invalid\n";
		}
		else if (com == 1){
			int aux; cin >> aux;
			num.push_back(aux);
			mapa[aux].push_back(num.size()-1);
		}
		else if (com == 2){
			int aux, i, j; cin >> aux >> i >> j;
			i--; j--;
			cout << lookFor(aux, i, j) << endl;

		}
	}

	return 0;
}
```

Teoria dos números

- Divisibility by 25

```
set<string> vis;
int bfs(string n){
	queue<pair<string, int> > q;
	q.push({n, 0});
	while(!q.empty()){
		pair<string, int> temp = q.front(); q.pop();
		//if contain leading zero
		if (temp.a[0] == '0') continue;
		//if already processed
		if (vis.count(temp.a) > 0) continue;
		vis.insert(temp.a);
		//if is div by 25 (00, 25, 50)
		if ((temp.a[int(temp.a.size())-2] == '0' && temp.a[int(temp.a.size())-1] ==
'0') ||
			(temp.a[int(temp.a.size())-2] == '2' &&
temp.a[int(temp.a.size())-1] == '5') ||
			(temp.a[int(temp.a.size())-2] == '5' &&
temp.a[int(temp.a.size())-1] == '0') )
			return temp.b;
		for (int i = 0; i < int(temp.a.size() - 1); i++){
			if (temp.a[i] == '0' || temp.a[i] == '2' || temp.a[i] == '5'){
				swap(temp.a[i], temp.a[i+1]);//go foward
				q.push({temp.a, 1+temp.b});
				swap(temp.a[i], temp.a[i+1]);//swap back
			}
		}
	}
	//if did not find any
	return -1;
}
int main(){
	ios::sync_with_stdio(false), cin.tie(0);
	string str;cin >> str;
	int zero=0, dois=0, cinco=0;
	for (int i = 0; i < int(str.size()); i++) if (str[i] == '7') str[i] = '2';
	for (auto x : str){
		if (x == '0') zero++;
		else if (x == '2') dois++;
		else if (x == '5') cinco++;
	}
	//se tem dois zeros, da (00)
	//se tem pelo menos um 0 ou um 2 & tem um cinco, consigo (25 ou 50)
	if (zero >=2 || ((zero || dois) && cinco)){
		int ans = bfs(str);
		if (ans == -1) cout << "-1\n";
		else cout << ans << endl;
	}else{
		cout << "-1\n";
	}
	return 0;}
```

SegTree

- Interval Products

```
vi cell(maxn), tree(4*maxn);
void build(int pos, int i, int j){
	int mid = (i+j)/2;
	int esq = pos*2;
	int dir = pos*2 + 1;
	if (i==j){
		tree[pos] = cell[i];
		return;
	}
	build(esq, i, mid);
	build(dir, mid+1, j);

	tree[pos] = tree[esq] * tree[dir];
}
int query(int pos, int i, int j, int l, int r){
	int mid = (i+j)/2;
	int esq = pos*2;
	int dir = pos*2 + 1;

	if (j < l || i > r){
		return 1;
	}
	if (i >= l && j <= r){
		//cout << pos << endl;
		return tree[pos];
	}

	return query(esq, i, mid, l, r)*query(dir, mid+1, j, l, r);
}
void update(int pos, int i, int j, int x, int value){
	int mid = (i+j)/2;
	int esq = pos*2;
	int dir = pos*2 + 1;
	if (x < i || x > j) return;

	if (i==j){
		tree[pos] = value;
		return;
	}
```

```
        update(esq, i, mid, x, value);
        update(dir, mid+1, j, x, value);

        tree[pos] = tree[esq] * tree[dir];
}

int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        int N, Q;
        while(cin >> N >> Q){
                for (int i = 0; i < N; i++){
                        cin >> cell[i];
                        if (cell[i]){
                                cell[i] /= abs(cell[i]);
                        }
                }
                build(1, 0, N-1);
                while(Q--){
                        char cmd; cin >> cmd;
                        if (cmd == 'P'){
                                int i, j; cin >> i >> j; i--;j--;
                                int aux = query(1, 0, N-1, i, j);
                                if (!aux) cout << "0";
                                else if (aux<0) cout << "-";
                                else cout << "+";
                        }else{
                                int x, v; cin >> x >> v; x--;
                                if (v)
                                        v /= abs(v);
                                update(1, 0, N-1, x, v);
                                cell[x] = v;
                        }
                }
                cout << endl;
        }
        return 0;
}
```

- Distinct Characters Queries

```
vector<pair<vi, int>> cell(maxn), tree(4*maxn);
void build(int pos, int i, int j){
	int mid = (i+j)/2;
	int esq = pos*2;
	int dir = pos*2 + 1;
	if (i==j){
		tree[pos] = cell[i];
		return;
	}
	build(esq, i, mid);
	build(dir, mid+1, j);

	int a = esq, b = dir;
	for (int i = 0; i < 26; i++){
		tree[pos].l[i] += tree[a].l[i] + tree[b].l[i];
	}
	tree[pos].s = 0;
	for (int i = 0; i < 26; i++){
		if (tree[pos].l[i])
			tree[pos].s++;
	}
}
pair<vi, int> query(int pos, int i, int j, int l, int r){
	int mid = (i+j)/2;
	int esq = pos*2;
	int dir = pos*2 + 1;
	if (j < l || i > r){
		pair<vi, int> aux;
		aux.l.resize(26,0);
		return aux;
	}
	if (i >= l && j <= r){
		return tree[pos];
	}
	pair<vi, int> L, R, neww;
	L = query(esq, i, mid, l, r);
	R = query(dir, mid+1, j, l, r);

	neww.l.resize(26,0);
	for (int i = 0; i < 26; i++){
			neww.l[i] += L.l[i] + R.l[i];
		}
		neww.s = 0;
		for (int i = 0; i < 26; i++){
			if (neww.l[i])
				neww.s++;
		}
		return neww;
}
void update(int pos, int i, int j, int x, int value, int removed){
		int mid = (i+j)/2;
		int esq = pos*2;
		int dir = pos*2 + 1;

		if (i==j){
			tree[pos].l[removed] = 0;
			tree[pos].l[value] = 1;
			tree[pos].s = 1;
			return;
		}
		tree[pos].l[removed] -= 1;
		if (tree[pos].l[removed] == 0)
			tree[pos].s -= 1;
		if (tree[pos].l[value] == 0)
			tree[pos].s += 1;
		tree[pos].l[value] += 1;

		if (x <= mid) update(esq, i, mid, x, value, removed);
		else update(dir, mid+1, j, x, value, removed);
}
int main(){
		ios::sync_with_stdio(false), cin.tie(0);
		string str; cin >> str;
		int Q; cin >> Q;
		int N = str.size();
		for (int i = 0; i < N; i++){
			pair<vi, int> aux; aux.l.resize(26,0);
			for (int i = 0; i < 26; i++) aux.l[i] = 0;
			aux.l[str[i]-97] = 1; aux.s = 1;
			cell[i+1] = aux;
		}
```

```
	int l = 0;
	for (auto x : tree){
		tree[l++].l.resize(26,0);
	}
	build(1, 1, N+1);

	while(Q--){
		int cmd; cin >> cmd;
		if (cmd == 2){
			int i, j; cin >> i >> j;
			pair<vi, int> aux = query(1, 1, N+1, i, j);
			cout << aux.s << endl;
		}else{
			int x; char c; cin >> x >> c;
			update(1, 1, N+1, x, c-97, str[x-1]-97);
			str[x-1] = c;
		}
	}
	return 0;
}
```

Strings

- Longest Palindromic Substring

```
ll pot[2][maxn], ahash[2][maxn];
int get_id(int i, int m){
	if (!m) return str1[i]-'a'+1;
	return str2[i]-'a'+1;
}
void build(int m){
	pot[m][0] = 1;
	ahash[m][0] = get_id(0, m);
	for (int i = 1; i < n; i++){
		pot[m][i] = (pot[m][i-1] * base) % mod;
		ahash[m][i] = ((ahash[m][i-1]*base) + get_id(i, m)) % mod;
	}
}
ll getkey(int l, int r, int m){
	ll res = ahash[m][r];
	if (l > 0) res = (res - ((pot[m][r-l+1] * ahash[m][l-1]) % mod) + mod) % mod;
	return res;
}
bool checkPal(int k){
        for (int i = 0; i <= n-k; i++){
                ll hash1 = getkey(i, i+k-1, 0);
                ll hash2 = getkey(n-i-k, n-1-i, 1);
                if (hash1 == hash2)
                        return true;
        }
        return false;
}
int par = 1, impar = 1;
vi vPar, vImp;
void bSearchP(int l, int r){
        while(l < r){
                int m = (l+r)/2;
                if(!checkPal(vPar[m])){
                        r = m;
                }else{
                        par = max(par, vPar[m]);
                        l = m+1;
                }
        }
}
void bSearchI(int l, int r){
        while(l < r){
                int m = (l+r)/2;
                if(!checkPal(vImp[m])){
                        r = m;
                }else{
                        impar = max(impar, vImp[m]);
                        l = m+1;
                }
        }
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);

        cin >> n;
        cin >> str1; n = str1.size();
        str2 = "";
        for (auto x : str1) str2 = x + str2;
        if (str1 == str2){cout << n << endl; return 0;}
        build(0); build(1);
```

```
	for (int i = 1; i <= n; i++)
		(i%2) ? vImp.pb(i) : vPar.pb(i);
	bSearchP(0, vPar.size()-1);
	bSearchI(0, vImp.size()-1);

	cout << max(par, impar) << endl;
	return 0;
}
```

- The text splitting

```
int n, p, q;
ll gcd(ll a, ll b){
	while(b) a %= b, swap(a,b);
	return a;
}

ll gcd_ext(ll a, ll b, ll &x, ll&y){
	if (b==0){
		x = 1;
		y = 0;
		return a;
	}
	ll nx, ny;
	ll gc = gcd_ext(b, a%b, nx, ny);
	x = ny;
	y = nx - (a/b)*ny;

	return gc;
}

int main(){
	ios::sync_with_stdio(false), cin.tie(0);
	cin >> n >> p >> q; string str; cin >> str; n = str.size();
	vs out;
	ll x, y;
	int m = -1;
	if (n%p == 0) m = p;
	if (n%q == 0) m = q;
	for (int i = 0; m != -1 && i < n; i+=m)
		out.pb(str.substr(i, m));
	if (m ==-1){
		//Diofantina
```

```
			out.resize(0);
			ll mdc = gcd(p,q);
			if(n%mdc != 0){cout << "-1\n"; return 0;}
			gcd_ext(p, q, x, y);
			x *= n/mdc;
			y *= n/mdc;
			while(x<0){
				x+=q/mdc;
				y-=p/mdc;
			}
			while(y<0){
				y+=p/mdc;
				x-=q/mdc;
			}
			//cout << x << " " << y << endl;
			if(x<0 || y< 0 ){cout << "-1\n"; return 0;}
			int i = 0;
			for (int k = 0; k < x; k++){
				out.pb(str.substr(i,p));
				i+=p;
			}
			for (int k = 0; k < y; k++){
				out.pb(str.substr(i,q));
				i+=q;
			}
		}

		cout << out.size() << endl;
		for (auto x : out){
			cout << x << endl;
		}

		return 0;
}
```

- Query on strings

```
int trie[ms][sigma], terminal[ms], ter[ms], z;
int get_id(char c){
	return c - 'a';
}
void init(){
	memset(trie[0], -1, sizeof(trie[0]));
```

```
        z = 1;
}
void insert(string &p){
        int cur = 0;
        for (int i = 0; i < int(p.size()); i++){
                int id = get_id(p[i]);
                ter[cur]++;
                if (trie[cur][id] == -1){
                        memset(trie[z], -1, sizeof(trie[z]));
                        trie[cur][id] = z++;
                }
                cur = trie[cur][id];
        }
        terminal[cur]++;
        ter[cur]++;
}
int count(string &p){
        int cur = 0;
        for (int i = 0; i < int(p.size()); i++){
                int id = get_id(p[i]);
                if (trie[cur][id] == -1)
                        return false;
                cur = trie[cur][id];
        }
        return terminal[cur];
}
void remove(string &p){
        int cur = 0;
        for(int i = 0 ; i < int(p.size()) ; i++){
                int id = get_id(p[i]);
                ter[cur]--;
                cur = trie[cur][id];
        }
        terminal[cur]--;
        ter[cur]--;
}
bool exists = false;
void ans(int cur, int cnt, int l, int k){
        if(exists) return;
        if(l == cnt){
                if(ter[cur] >= k)
                exists = true;
            return;
        }
        if(ter[cur] < k) return;
        for(int id = 0 ; id < 26 ; id++){
            if(trie[cur][id] != -1){
                ans(trie[cur][id], cnt+1, l, k);
            }
        }
}
int main(){
        ios::sync_with_stdio(false), cin.tie(0);
        init();
        int n; cin >> n;
        vs words(n+5);
        vi there(n+5);
        for (int i = 1; i <= n; i++){
            string cmd; cin >> cmd;
            if (cmd == "1"){
                string str; cin >> str;
                reverse(str.begin(), str.end());
                there[i] = true;
                words[i] = str;
                insert(str);
            }
            else if (cmd == "3"){
                int ind; cin >> ind;
                if (there[ind])
                    remove(words[ind]);
                there[ind] = false;
            }
            else{
                int k, l; cin >> k >> l;
                exists = false;
                ans(0, 0, l, k);
                cout << ((exists) ? "YES" : "NO") << endl;
            }
        }
        return 0;
}
```