Contents

1 기초적인 코드 조각들

1.1 최댓값/위치 찾기

```
int arr[10] = {5,1,2,5,7,4,1,2,5,2};
int len = 10;
int mxidx = 0;
for(int i = 0; i < len; i++){
  if(arr[i] > arr[mxidx]) mxidx = i;
}
cout << "max idx : " << mxidx << '\n';
cout << "max val : " << arr[mxidx] << '\n';
/*
max idx : 4
max val : 7
*/
```

1.2 배열 중간에 삽입

```
// val을 0-indexed 기준 arr의 idx번째에 추가
// ex : insert({0 1 2 3 5 6}, 6, 9, 6)
//           -> 0 1 2 3 9 5 6
void insert(int* arr, int& len, int& val, int& idx){
  for(int i = len-1; i >= idx; i--) arr[i+1] = arr[i];
  arr[idx] = val;
  len++;
}
```

1.3 배열 중간에 제거

```
// 0-indexed 기준 arr의 idx번째 원소를 제거
// ex : delete({0 1 2 3 7 6}, 2)
//           -> 0 1 3 7 6
void erase(int* arr, int& len, int& idx){
  for(int i = idx; i < len-1; i++) arr[i] = arr[i+1];
  len--;
}
```

1.4 각종 경우의 수

```
// 순서 없이 5개에서 3개 뽑기(조합)
int arr1[5] = {0, 0, 1, 1, 1};
do{
  for(int i = 0; i < 5; i++)
    if(arr1[i] == 1) cout << i+1 << ' ';
  cout << '\n';
}while(next_permutation(arr1, arr1+5));
// (345), (245), (235), (234), ...

// 순서 있게 5개에서 3개 뽑기(순열)
int arr2[5] = {0, 0, 1, 2, 3};
do{
  int seq[3];
  for(int i = 0; i < 5; i++)
    if(arr2[i] != 0) seq[arr2[i]-1] = i+1;
  for(int i = 0; i < 3; i++)
    cout << seq[i] << ' ';
  cout << '\n';
} while(next_permutation(arr2,arr2+5));
// (345), (354), (435), (534), ...

// 순서 없이 중복 허용해 5개에서 3개 뽑기(중복조합)
int seq[3];
void solve(int cur, int depth){
  if(depth == 3){
    for(int i = 0; i < 3; i++)
      cout << seq[i] << ' ';
    cout << '\n';
    return;
  }
  for(int i = cur; i <= 5; i++){
    seq[depth] = i;
    solve(i, depth+1);
  }
}
solve(1, 0);
// (111), (112), (113), (114), (115), (122), ...
```

```cpp
// 순서 있게 중복 허용해 5개에서 3개 뽑기(중복순열)
int mask = 5*5*5; // 5^3
for(int mask = 0; mask < 5*5*5; mask++){
  int tmp = mask;
  for(int i = 0; i < 3; i++){
    cout << tmp % 5 + 1;
    tmp /= 5;
  }
  cout << '\n';
}
// (111), (211), (311), (411), (511), (121), ...
```

1.5 BFS

```cpp
#define X first
#define Y second // pair에서 first, second를 줄여서 쓰기 위해서 사용
int board[502][502] =
{{1,1,1,0,1,0,0,0,0,0},
 {1,0,0,0,1,0,0,0,0,0},
 {1,1,1,0,1,0,0,0,0,0},
 {1,1,0,0,1,0,0,0,0,0},
 {0,1,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0} }; // 1이 파란 칸, 0이 빨간 칸에 대응
bool vis[502][502]; // 해당 칸을 방문했는지 여부를 저장
int n = 7, m = 10; // n = 행의 수, m = 열의 수
int dx[4] = {1,0,-1,0};
int dy[4] = {0,1,0,-1}; // 상하좌우 네 방향을 의미
int BFS(){
  queue<pair<int,int> > Q;
  vis[0][0] = 1; // (0, 0)을 방문했다고 명시
  Q.push({0,0}); // 큐에 시작점인 (0, 0)을 삽입.
  while(!Q.empty()){
    auto cur = Q.front(); Q.pop();
    cout << '(' << cur.X << ", " << cur.Y << ") -> ";
    for(int dir = 0; dir < 4; dir++){ // 상하좌우 칸을 살펴볼 것이다.
      int nx = cur.X + dx[dir];
      int ny = cur.Y + dy[dir]; // nx, ny에 dir에서 정한 방향의 인접한 칸의 좌표가 들어감
      if(nx < 0 or nx >= n or ny < 0 or ny >= m) continue; // 범위 밖일 경우 넘어감
      if(vis[nx][ny] or board[nx][ny] != 1) continue; // 이미 방문한 칸이거나 파란 칸이
 ↪  아닐 경우
      vis[nx][ny] = 1; // (nx, ny)를 방문했다고 명시
      Q.push({nx,ny});
    }
  }
}
// (0, 0) -> (1, 0) -> (0, 1) -> (2, 0) -> (0, 2)...
```

2 시험환경에 관한 정보

2.1 C++11 미지원

2019-09-06 기준 지금까지 삼성 역량테스트 A형에서는 C++11을 지원하지 않았고 이번 시험에도 지원하지 않을 가능성이 큼. 그로 인해 로컬 Visual Studio 에서는 실행이 잘 되나 서버에 제출시 오류를 낼 수 있는 아래 사항들에 유의.

- $vector < pair < int, int >> \rightarrow vector < pair < int, int > >$ 으로 수정해야 함 (>과 > 사이 띄어쓰기)
- tuple 사용 불가
- 중괄호를 이용한 초기화 불가 ($pair < int, int > t = \{1, 2\}$가 에러, $make_pair(1, 2)$를 사용해야 함)
- auto 불가

2.2 Visual Studio 2017 관련 정보

자주 발생하는 오류 메시지

- Error C2872 : ambiguous symbol (모호한 기호입니다), Error C2365 :redefinition; previous definition was 'function'(재정의: 이전 정의는 '함수'입니다.) → std namespace 내에 이미 정의되어 있는 변수/함수/클래스의 이름과 동일한 변수를 선언해서 발생하는 에러, 변수명을 수정해서 해결
- Error C2065 : undeclared identifier (선언되지 않은 식별자입니다.) → 해당 변수가 선언 전에 사용됨. 선언을 먼저 해야 함
- Error C4996 : This function or variable may be unsafe → #define _CRT_SECURE_NO_WARNINGS 혹은 #pragma warning(disable : 4996) 를 코드 상단에 삽입

디버깅 팁

알고리즘 문제를 풀 땐 길어봐야 200줄 안의 짧은 프로그램이고 시간이 촉박하기에 디버그 모드로 디버깅을 하는 대신 콘솔 출력으로 오류를 찾는 것이 좋음
그러나 런타임 에러가 발생하는 입력을 발견했는데 정확히 코드 상의 어느 줄에서 발생했는지를 모를 땐 Visual Studio의 디버그 모드를 활용하면 빠르게 찾을 수 있음. 이를 위해 디버그 모드(F5)로 실행해야 함.

3 자주 실수하는 점 및 코딩 팁

- 반드시 최대 크기의 입력을 넣어서 시간 내로 잘 나오는지, 배열 크기 문제는 없는지 확인
- N이 0 혹은 1과 같은 최소 케이스에서 문제가 없는지 확인
- 배열 크기는 런타임에러가 나지 않도록 넉넉하게 잡기
- for문이 중첩되어 있을 때 변수명 꼬이지 않게 주의
- 각 테스트케이스가 종료된 후에 사용한 변수들을 초기화해야 함. 코드를 눈으로 보는 것 뿐만 아니라 직접 작은 테스트케이스 뒤에 큰 테스트케이스를 넣는다거나, 주어진 테스트케이스를 5번 정도 붙여넣어본다거나 하는 방법으로 답이 잘 나오는 것을 확인해야 함
- 함수로 인자를 주고 받을 때 변수 자체가 넘어간 것인지 주소가 넘어간 것인지 주의
- 문제에 주어진 조건을 단 한개라도 놓치지 않게 주의. 모든 조건을 따로 메모장이나 종이에 적어두는 것도 괜찮음
- 문제에서 주어져있지 않은 조건을 임의로 가정하지 않도록 주의
- A형의 경우 일단 2문제를 모두 읽고 구현이 쉬운 문제부터 먼저 시도하기
- 구상을 99%까지는 완벽하게 해두고 구현에 들어가기
- 재귀가 필요할 경우 입력값, 반환값, 탈출 조건을 매우매우 세밀하게 잘 잡아두어야 함
- BFS에서 시간초과 혹은 런타임에러 발생시 방문 체크 루틴을 확인
- 짜다가 꼬였을 경우 계속 달려들어 스파게티 코드로 만드는 것 보다는 5분 정도 머릿속을 정리하며 문제가 생긴 부분이 빠르게 해결 가능한 점인지 냉철하게 판단하고 이를 바탕으로 지금 보는 문제를 과감하게 버릴지, 계속 붙잡을지 판단