

2024 겨울학기 동국대학교SW역량강화캠프

3일차. 반복문을 활용한 완전탐색 II

### ● 완전탐색

- ▶ 가능한 모든 경우의 수를 고려하는 가장 기본적인 탐색 알고리즘
- ▶ 가장 무식하지만 가장 확실한 방법. 답을 찾을 확률 100%.
- ▶ 주로 반복문, DFS(깊이 우선 탐색), BFS(너비 우선 탐색)을 사용

● 완전 탐색 문제 유형 - 기준 잡아 비교하기

- ▶ 주어진 값들 중 하나를 잡아, 나머지와 비교하여 계산하는 형태의 문제

## ● 덩치 (767)

### 문제

우리는 사람의 덩치를 키와 몸무게, 이 두 개의 값으로 표현하여 그 등수를 매겨보려고 한다. 어떤 사람의 몸무게가  $x$  kg이고 키가  $y$  cm라면 이 사람의 덩치는  $(x,y)$ 로 표시된다. 두 사람 A 와 B의 덩치가 각각  $(x,y)$ ,  $(p,q)$ 라고 할 때  $x > p$  그리고  $y > q$  이라면 우리는 A의 덩치가 B의 덩치보다 "더 크다"고 말한다. 예를 들어 어떤 A, B 두 사람의 덩치가 각각  $(56,177)$ ,  $(45,165)$  라고 한다면 A의 덩치가 B보다 큰 셈이 된다. 그런데 서로 다른 덩치끼리 크기를 정할 수 없는 경우도 있다. 예를 들어 두 사람 C와 D의 덩치가 각각  $(45, 181)$ ,  $(55,173)$ 이라면 몸무게는 D가 C보다 더 무겁고, 키는 C가 더 크므로, "덩치"로만 볼 때 C와 D는 누구도 상대방보다 더 크다고 말할 수 없다.

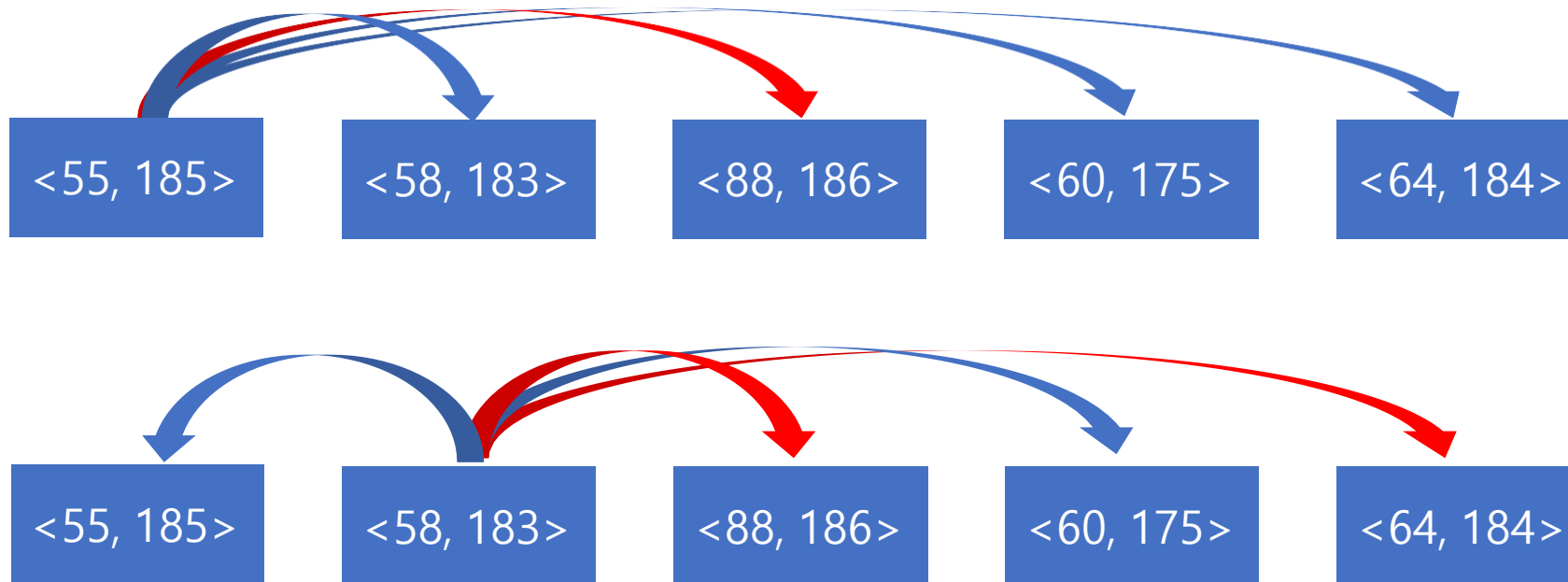
N명의 집단에서 각 사람의 덩치 등수는 자신보다 더 "큰 덩치"의 사람의 수로 정해진다. 만일 자신보다 더 큰 덩치의 사람이 k명이라면 그 사람의 덩치 등수는  $k+1$ 이 된다. 이렇게 등수를 결정하면 같은 덩치 등수를 가진 사람은 여러 명도 가능하다. 아래는 5명으로 이루어진 집단에서 각 사람의 덩치와 그 등수가 표시된 표이다

이름	<몸무게, 키>	덩치 등수
A	<55, 185>	2
B	<58, 183>	2
C	<88, 186>	1
D	<60, 175>	2
E	<46, 155>	5

위 표에서 C보다 더 큰 덩치의 사람이 없으므로 C는 1등이 된다. 그리고 A, B, D 각각의 덩치보다 큰 사람은 C뿐이므로 이들은 모두 2등이 된다. 그리고 E보다 큰 덩치는 A, B, C, D 이렇게 4명이므로 E의 덩치는 5등이 된다. 위 경우에 3등과 4등은 존재하지 않는다. 여러분은 학생 N명의 몸무게와 키가 담긴 입력을 읽어서 각 사람의 덩치 등수를 계산하여 출력해야 한다.

### ● 완전탐색을 통한 해결

▶ 모든 사람에 대해서, 자신보다 덩치가 큰 사람이 몇 명인지 계산하면 된다.



```
for (int i = 0; i < N; i++) {  
    int bigger = 0; // i보다 덩치가 큰 사람을 세어준다.  
    for (int j = 0; j < N; j++) {  
        if (i == j)  
            continue; // 자기 자신은 제외  
        if (arr[i][0] < arr[j][0] && arr[i][1] < arr[j][1]) { // j가 i보다 키와 몸무게 둘 다 크다면  
            bigger++;  
        }  
    }  
    sb.append((bigger + 1) + " "); // 자신보다 덩치가 큰 사람의 수 + 1 을 출력  
}
```

## ● 화살표그리기 (553)

### 문제

직선 위에 위치를 나타내는 0, 1, 2, ...와 같은 음수가 아닌 정수들이 일정한 간격으로 오른쪽 방향으로 놓여 있다. 이러한 위치들 중  $N$ 개의 위치에 하나씩 점들이 주어진다(<그림 1>). 주어진 점들의 위치는 모두 다르다. 두 점 사이의 거리는 두 점의 위치를 나타내는 수들의 차이이다. <그림 1>에서는 4개의 점이 주어지고 점  $a$ 와  $b$ 의 거리는 3이다.

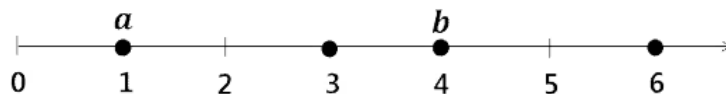


그림1

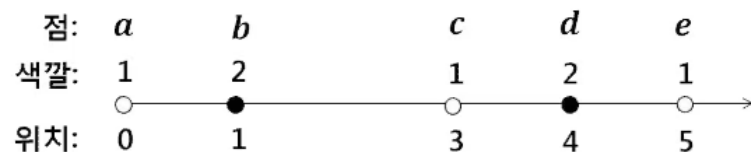
각 점은  $N$ 개의 색깔 중 하나를 가진다. 편의상, 색깔은 1부터  $N$ 까지의 수로 표시한다.

각 점  $p$ 에 대해서,  $p$ 에서 시작하는 직선 화살표를 이용해서 다른 점  $q$ 에 연결하려고 한다. 여기서, 점  $q$ 는  $p$ 와 같은 색깔의 점들 중  $p$ 와 거리가 가장 가까운 점이어야 한다. 만약 가장 가까운 점이 두 개 이상이면 아무거나 하나를 선택한다.

모든 점에 대해서 같은 색깔을 가진 다른 점이 항상 존재한다. 따라서 각 점  $p$ 에서 시작하여 위 조건을 만족하는  $q$ 로 가는 하나의 화살표를 항상 그릴 수 있다.

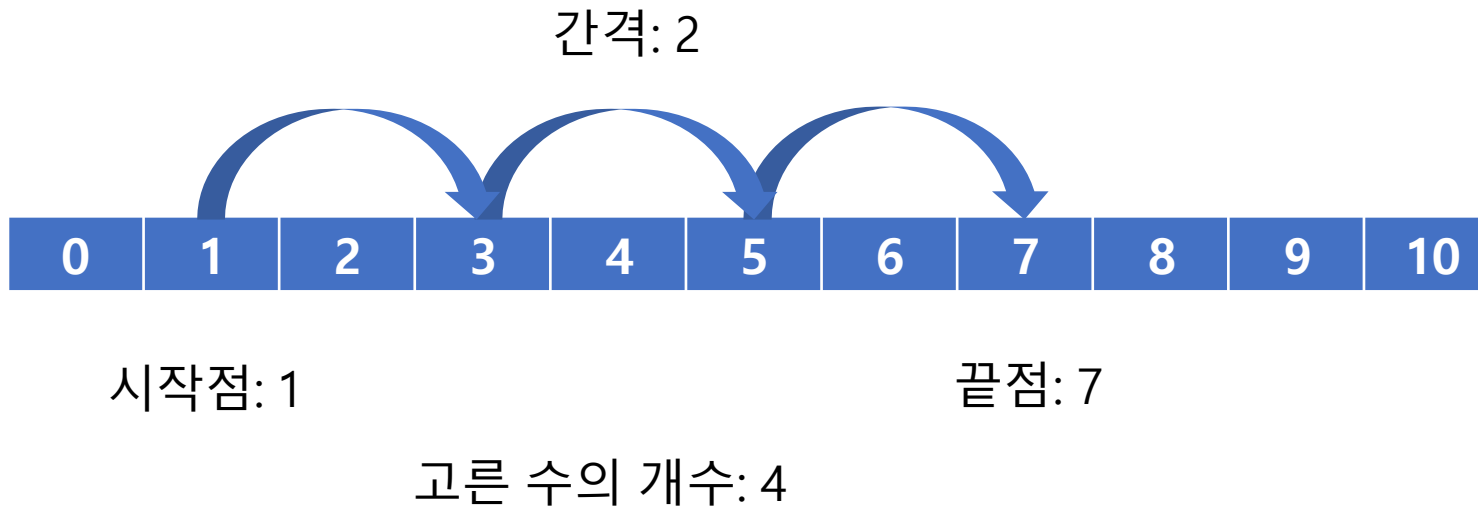
예를 들어, 점들을 순서쌍 (위치, 색깔)로 표시할 때,  $a = (0, 1)$ ,  $b = (1, 2)$ ,  $c = (3, 1)$ ,  $d = (4, 2)$ ,  $e = (5, 1)$ 라고 하자.

아래 <그림 2>에서 이 점들을 표시한다. 여기서 흰색은 1, 검은색은 2에 해당된다.



## ● 완전 탐색 문제 유형 - 구간 탐색

- ▶ 배열에서 특정 조건을 만족하는 구간을 찾는 형태의 문제
- ▶ 4가지 변수를 조절하면서 반복문을 돌리며, 구간을 찾는다.
- ▶ 구간의 시작점, 구간의 끝점, 고르는 수들 사이의 간격, 고르는 수의 개수





- 시작과 끝의 위치를 정하는 경우
  - 끝의 위치를 어디로 정하느냐도 중요
    1. 구간의 정확히 끝 지점을  $e$ 로 잡는다.
      - 직관적이다.
      - $e$ 를 관리하는 for문을 작성하기 편하다
    2. 구간의 끝 지점 + 1칸을  $e$ 로 잡는다
      - python에서 `range(s, e)`를 쓰기 편하다.
      - $(e - s)$ 로 쉽게 구간의 길이를 구할 수 있다.

## 오늘 배울 개념

- 시작과 끝의 위치를 정하는 경우
  - 끝의 위치를 어디로 정하느냐도 중요
    1. 구간의 정확히 끝 지점을 e로 잡는다.

```
for(s=0; s<n; s++){  
    for(e=s; e<n; e++){  
        for(i=s; i<=e; i++){  
            // 구간의 길이 = e - s + 1
```

2. 구간의 끝 지점 + 1칸을 e로 잡는다

```
for(s=0; s<n; s++){  
    for(e=s+1; e<=n; e++){  
        for(i=s; i<e; i++){  
            // 구간의 길이 = e - s
```

## 오늘 배울 개념

- 시작 위치와 길이를 정하는 경우

```
for(s=0; s<n; s++){  
    for(l=1; l<n; l++){  
        if(s+l >= n) break;  
        for(i=s; i<s+l; i++){  
            arr[i] = ...  
        }  
    }  
}
```

```
for(s=0; s<n; s++){  
    for(l=1; l<n; l++){  
        if(s+l >= n) break;  
        for(i=0; i<l; i++){  
            arr[s+i] = ...  
        }  
    }  
}
```

## 오늘 배울 개념

- 시작과 끝의 위치, 간격을 정하는 경우

```
for(s=0; s<n; s++){  
    for(e=s+1; e<=n; e++){  
        for(d=1; d<n; d++){  
            for(i=s; i<e; i+=d){  
                arr[i] = ...  
            }  
        }  
    }  
}
```

## 오늘 배울 개념

- 시작의 위치와 길이, 간격을 정하는 경우
  - 길이를 어떻게 정의하느냐도 중요
    1. 길이를 가져올 대상의 길이로 지정 (간격에 따라 가져오는 값의 개수가 달라짐)

```
for(s=0; s<n; s++){  
    for(l=1; l<=n; l++){  
        if(s+l >= n) break;  
        for(d=1; d<n; d++){  
            for(i=s; i<s+l; i+=d){  
                arr[i] = ...  
            }  
        }  
    }  
}
```

2. 길이를 가져올 대상의 개수로 지정

```
for(s=0; s<n; s++){  
    for(l=1; l<=n; l++){  
        for(d=1; d<n; d++){  
            if(s + (l-1)*d >= n) break;  
            for(i=0; i<l; i++){  
                arr[s + i*d] = ...  
            }  
        }  
    }  
}
```

길이나 끝점의 제한이 없는 경우는  $l = n - s$ ,  $e = n$  이라고 두고 코딩을 해도 무관함

나머지 응용은 알아서...

## ● 표지 (4103)

### | 문제

윌리는 새로운 표지를 제작하려고 한다.

새 표지를 만드는 일은 굉장히 비용이 많이들어 기존에 있는 표지를 재활용하려고 한다.

표지들은 알파벳 소문자로 구성되어 있다.

기존 표지로 새 표지를 만들기 위해서는, 기존 표지에서 같은 간격으로 있는 문자를 선택해서 새로운 표지 단어와 정확히 일치해야한다.

예를 들어, 만들고자 하는 새로운 표지가 abc 이고, 기존 표지가 aabc 라면, 2, 3, 4번째 문자를 선택하여 새로운 표지를 만들 수 있다.

하지만, 기존 표지가 abdc 라면, 1, 2, 4번째 문자가 같은 간격으로 존재하지 않으므로 새로운 표지로 만들 수 없다.

$N$ 개의 기존 표지와 새로운 표지가 주어졌을 때, 새로운 표지로 만들 수 있는 기존 표지의 개수를 구하자.

### | 입력

첫째 줄에 기존 표지의 수  $N$ 이 주어진다. ( $1 \leq N \leq 100$ ) 둘째 줄에는 윌리의 새 표지 이름이 주어진다. 이름은 알파벳 소문자로만 이루어져 있고, 길이는 3자 이상, 25자 이하이다. 다음  $N$ 개 줄에는 기존 표지에 쓰여 있는 문자가 주어진다. 이 표지에 쓰여 있는 문자는 알파벳 소문자이고, 길이는 1자 이상 100자 이하이다.

### | 출력

첫째 줄에 상근이가 만들 수 있는 표지의 수를 출력한다.

### ●반복문을 이용한 완전탐색 풀이

- ▶ 문자열 str에서 조건대로 문자를 골라 board 문자열을 완성할 수 있는지 탐색
- ▶ str에서 시작점과 고르는 문자들 사이의 간격을 결정해서 문자들을 고른다.
- ▶ str에서 시작점 j, 간격 d로 len개의 문자를 고르면

$\text{str}[j], \text{str}[j+d], \text{str}[j+2*d], \dots, \text{str}[j+(len-1)*d]$



```
cin >> n;
cin >> target;

while(n--){
    cin >> str;

    bool pos=false;
    for(s=0; s<str.size(); s++){
        for(d=1; d<str.size(); d++){
            if(s+d*(target.size()-1) >= str.size()) continue;

            flag = true;
            for(k=0; k<target.size(); k++){
                if(target[k] != str[s+k*d]) flag = false;
            }

            if(flag) pos = true;
        }
    }

    if(pos) ans++;
}

cout << ans;
```

## ● 평균이 들어있는 구간 구하기 (4093)

### 문제

윌리는  $N$ 개의 숫자에 대해서 아래 조건을 만족하는 구간의 개수를 구하려 한다.

윌리는  $(i, j)$ 구간을 선택하려고 하며  $(1 \leq i \leq j \leq N)$  이 구간은  $i$ 와  $j$ 를 포함하는 구간이다.

이 구간에 포함된 수들의 평균을 "평균 수"라고 하자.

$(i, j)$ 구간 안에 "평균 수" 가 들어있다면 조건을 만족하는 구간이고, 들어있지 않다면 조건을 만족하지 않는 구간이다.

위 조건을 만족하는 구간의 수를 출력하자.

첫 번째 예제에서는  $(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 4)$  구간이 조건을 만족한다.

### 입력

첫째줄에 수의 개수를 나타내는 양의 정수  $N$ 이 주어진다.  $1 \leq N \leq 100$

둘째 줄에는 배열을 구성하고 있는 양의 정수  $a_i$ 가  $N$ 개 주어진다.  $1 \leq a_i \leq 1000$

### 출력

조건을 만족하는 구간의 수를 출력하자.

### ● 체스판 색칠놀이 (4096)

#### 문제

윌리는  $M \times N$ 크기의 보드판을 가지고 있다.

보드의 각 칸은 흰색으로 색칠되어 있거나 검은색으로 색칠되어 있으며 이 보드를 잘라서  $8 \times 8$ 크기의 체스판을 만드려고 한다.

체스판은 흰색과 검은색이 번갈아 나타나는 형태이며, 왼쪽 위 칸의 색이 흰색일 수도 있고 검은색일 수도 있다.

보드판을  $8 \times 8$ 형태로 자른 후에 바로 체스판이 만들어지지 않는다면, 체스판으로 만들기 위해 몇 개의 칸을 다른 색으로 덧칠해야만 한다.

보드판에서  $8 \times 8$ 크기는 아무렇게나 고를 수 있다고 할 때, 다시 칠하는 칸의 수를 최소화하는 프로그램을 만들어보자.

#### 입력

첫 번째 줄에는 양의 정수  $N$ 과  $M$ 이 주어진다.  $8 \leq N, M \leq 50$

두 번째 줄부터  $N$ 개 줄에는 보드판의 상태가 주어진다.

흰 색으로 색칠되어 있다면 'W'로, 검은 색으로 색칠되어 있다면 'B'로 입력된다.

#### 출력

다시 칠해야 하는 정사각형 수의 최소값을 출력하자.

### ●반복문을 이용한 완전탐색 풀이

- ▶ 시작점이 정해지면 살펴봐야 하는 보드판의 범위가 결정된다
  - ▶ 시작점  $(x, y)$ 를 완전 탐색으로 결정한다
- ▶ 체스판의 성질: 같은 색을 가진 두 좌표의  $(x+y)$ 의 홀짝성이 같다
- ▶  $(x+y)$ 가 짝수인 칸에 'B'가 있고,  $(x+y)$ 가 홀수인 칸에 'W'가 있는지 개수 세기
- ▶  $(x+y)$ 가 짝수인 칸에 'W'가 있고,  $(x+y)$ 가 홀수인 칸에 'B'가 있는지 개수 세기

## 핵심 코드

```
cin >> n >> m;
for(i=0; i<n; i++){
    cin >> board[i];
}

for(i=0; i<=n-8; i++){
    for(j=0; j<=m-8; j++){
        cnt1 = cnt2 = 0;
        for(x=i; x<i+8; x++){
            for(y=j; y<j+8; y++){
                if((x+y)%2 == 0){
                    if(board[x][y] == 'B') cnt1++;
                    if(board[x][y] == 'W') cnt2++;
                }
                if((x+y)%2 == 1){
                    if(board[x][y] == 'B') cnt2++;
                    if(board[x][y] == 'W') cnt1++;
                }
            }
        }

        ans = min(ans, min(cnt1, cnt2));
    }
}

cout << ans;
```

## 핵심 코드

```
int n, m, cnt, ans=64, i, j, ii, jj;
int board[50][50];
char c;

cin >> n >> m;
for(i=0; i<n; i++){
    for(j=0; j<m; j++){
        cin >> c;
        board[i][j] = c == 'B';
    }
}

for(i=0; i<=n-8; i++){
    for(j=0; j<=m-8; j++){
        cnt = 0;
        for(x=i; x<i+8; x++){
            for(y=j; y<j+8; y++){
                cnt += board[x][y] == (x+y)%2;
            }
        }

        ans = min(ans, min(cnt, 64-cnt));
    }
}

cout << ans;
```

## ● 정사각형 찾기 (296)

### 문제

윌리는  $N$ 개의 숫자에 대해서 아래 조건을 만족하는 구간의 개수를 구하려 한다.

윌리는  $(i, j)$ 구간을 선택하려고 하며  $(1 \leq i \leq j \leq N)$  이 구간은  $i$ 와  $j$ 를 포함하는 구간이다.

이 구간에 포함된 수들의 평균을 "평균 수"라고 하자.

$(i, j)$ 구간 안에 "평균 수" 가 들어있다면 조건을 만족하는 구간이고, 들어있지 않다면 조건을 만족하지 않는 구간이다.

위 조건을 만족하는 구간의 수를 출력하자.

첫 번째 예제에서는  $(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 4)$  구간이 조건을 만족한다.

### 입력

첫째줄에 수의 개수를 나타내는 양의 정수  $N$ 이 주어진다.  $1 \leq N \leq 100$

둘째 줄에는 배열을 구성하고 있는 양의 정수  $a_i$ 가  $N$ 개 주어진다.  $1 \leq a_i \leq 1000$

### 출력

조건을 만족하는 구간의 수를 출력하자.

### ● 마라톤 (3370)

#### 문제

농장에 있는 젖소들이 건강하지 못하다고 생각한 농부 존은 젖소들을 위한 마라톤 대회를 열었고, 농부 존의 총애를 받는 젖소 박승원 역시 이 대회에 참가할 예정이다.

마라톤 코스는  $N$  ( $3 \leq N \leq 100000$ ) 개의 체크포인트로 구성되어 있으며, 1번 체크포인트에서 시작해서 모든 체크 포인트를 순서대로 방문한 후  $N$ 번 체크포인트에서 끝나야지 마라톤이 끝난다. 게으른 젖소 박승원은 막상 대회에 참가하려 하니 귀찮아져서 중간에 있는 체크포인트 한개를 몰래 건너뛰려 한다. 단, 1번 체크포인트와  $N$ 번 체크포인트를 건너뛰면 너무 눈치가 보이니 두 체크포인트는 건너뛰지 않을 생각이다.

젖소 박승원이 체크포인트 한개를 건너뛰면서 달릴 수 있다면, 과연 승원이가 달려야 하는 최소 거리는 얼마일까?

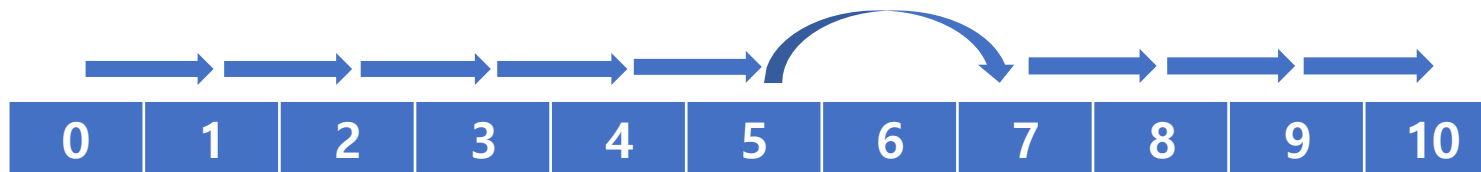
참고로, 젖소 마라톤 대회는 서울시내 한복판에서 진행될 예정이기 때문에 거리는 택시 거리(Manhattan Distance)로 계산하려고 한다. 즉,  $(x_1, y_1)$ 과  $(x_2, y_2)$  점 간의 거리는  $|x_1 - x_2| + |y_1 - y_2|$  로 표시할 수 있다. ( $|x|$ 는 절댓값 기호다.)

예를 들어 예시에서 젖소 박승원은 2번째 혹은 3번째 체크포인트를 건너뛸 수 있는데, 여기서 두 번째 체크포인트를 건너뛸 경우 경로는  $(0,0) \rightarrow (11,-1) \rightarrow (10,0)$  이 되며 거리는 14이다. 박승원은 이것보다 더 짧게 달릴 수 없다.



●반복문을 이용한 완전탐색 풀이  $O(N^2)$

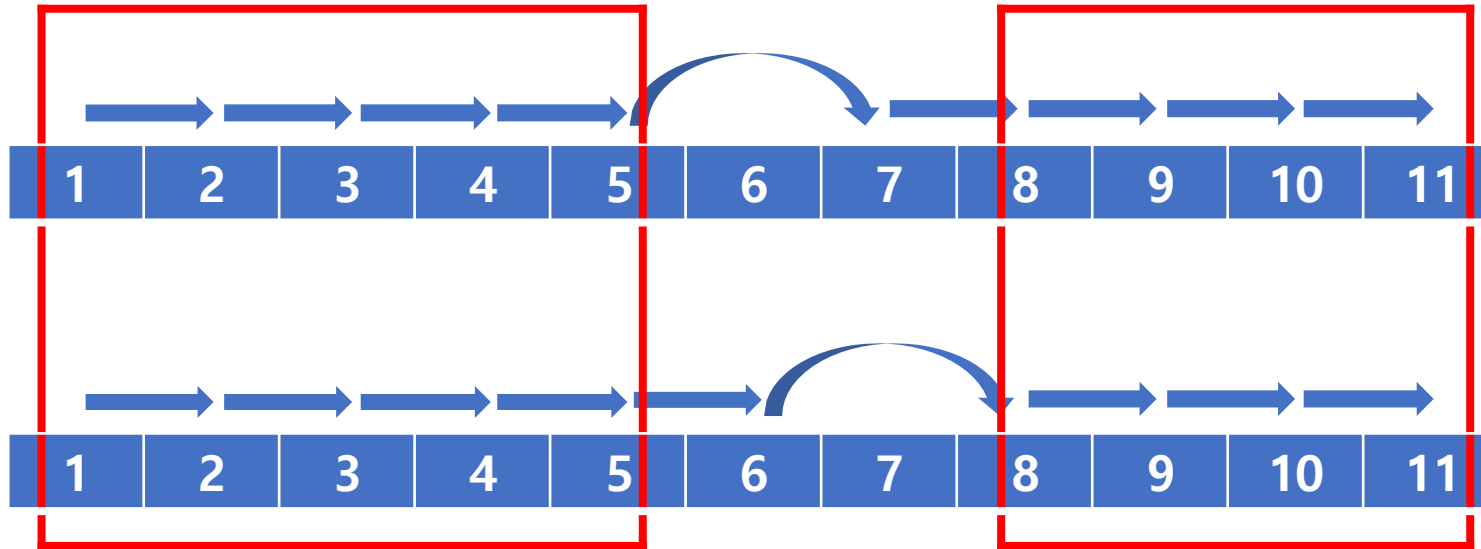
▶ 어느 체크포인트를 건너뛸 지를 완전탐색



```
int ans = 1987654321;
for(int i=2; i<N; i++) { // i번 체크포인트를 건너뛰어보자
    int totaldist = 0;
    for(int j=1; j<N; j++) { // j ~ (j+1) 거리를 더해준다.
        if(j == i - 1) { // 만약 j+1번을 건너뛰어야 한다면
            totaldist += dist(j, j+2); // j번과 j+2번 사이의 거리를 더해준다.
        }
        else if(j == i) { // j번을 건너뛰어야 한다면
            continue; // 어떠한 값도 더해주지 않는다.
        }
        else { // j ~ (j+1) 거리를 더해준다.
            totaldist += dist(j, j+1);
        }
    }
    if(totaldist < ans) ans = totaldist;
}
```

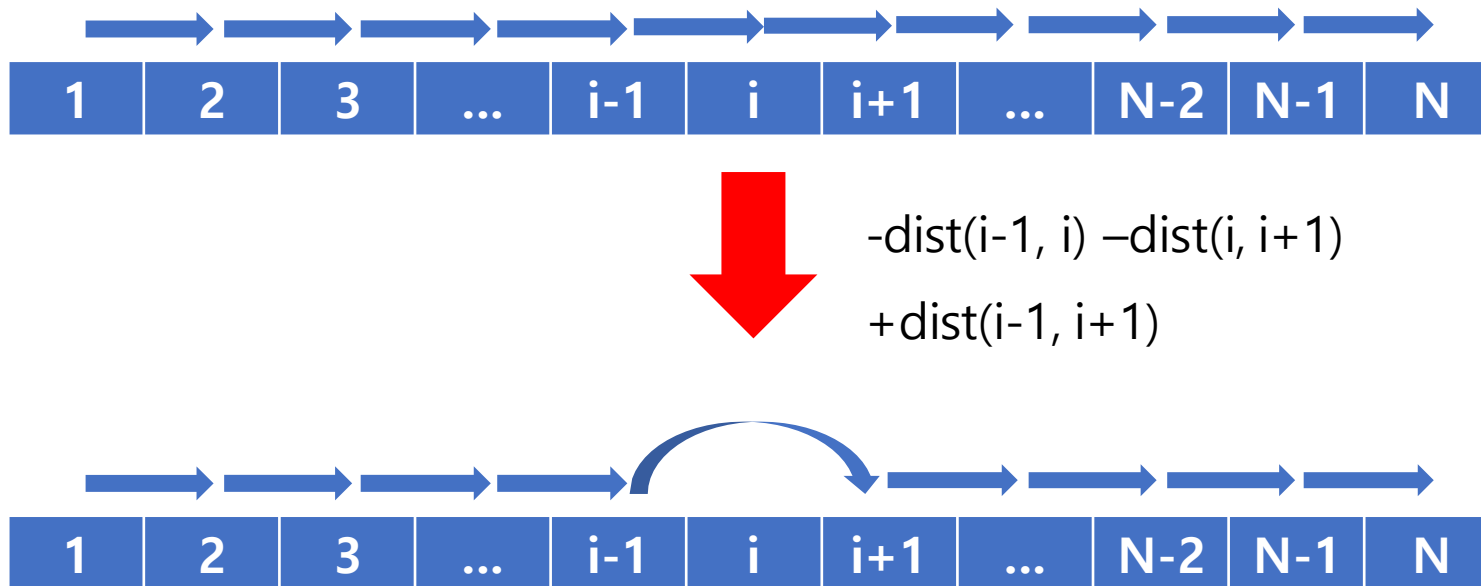
### ●반복문을 이용한 완전탐색 풀이 $O(N)$

- ▶ 어느 체크포인트를 건너뛸 지를 완전탐색
- ▶ 시간복잡도를 줄일 방법은?



### ●반복문을 이용한 완전탐색 풀이 $O(N)$

- ▶ 중복되는 계산을 줄이자.
- ▶ 체크포인트를 건너뛰지 않은 상황에서의 변화값만을 계산



```
int basedist = 0;
for(int i=2; i<=N; i++) {
    basedist += dist(i-1, i);
}
int ans = 1987654321;
for(int i=2; i<N; i++) { // i번 체크포인트를 건너뛰어보자
    int totaldist = basedist - dist(i-1, i) - dist(i, i+1) + dist(i-1, i+1);
    if(totaldist < ans) ans = totaldist;
}
```

## ● 회전 초밥 (811)

### 문제

회전 초밥 음식점에는 회전하는 벨트 위에 여러 가지 종류의 초밥이 놓여 있다.  
초밥의 종류를 번호로 표현할 때, 다음 그림은 회전 초밥 음식점의 벨트 상태의 예를 보여주고 있다.  
벨트 위에는 같은 종류의 초밥이 둘 이상 있을 수 있다.

