Day4 그리디, 분할 정복

선린인터넷고등학교 소프트웨어과
30610 나정휘
https://JusticeHui.github.io

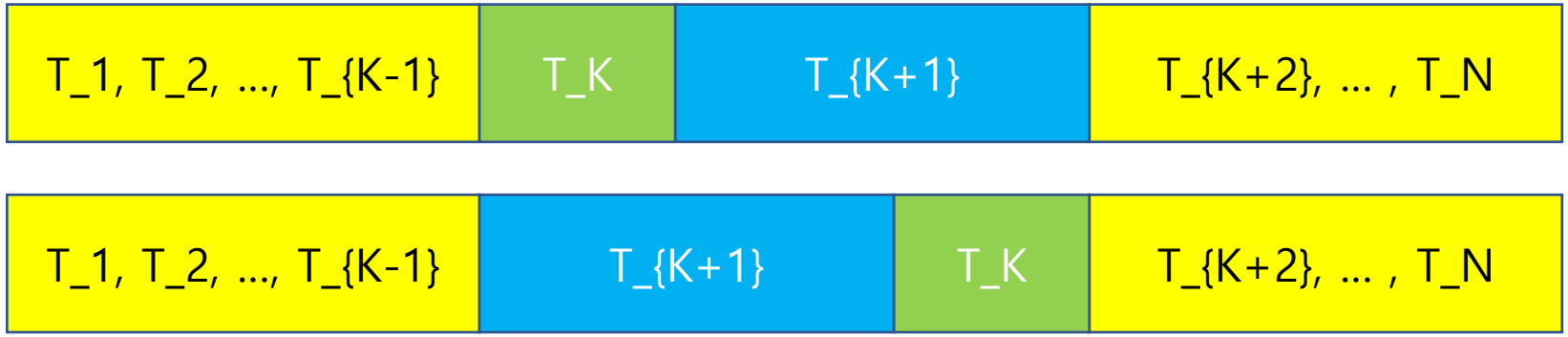
문제 목록

- BOJ14908 구두 수선공
- BOJ1992 쿼드트리
- BOJ11819 The Shortest does not Mean the Simplest
- BOJ6597 트리 복구
- BOJ10090 Counting Inversions
- BOJ11877 홍수
- BOJ2878 캔디캔디
- BOJ16288 Passport Control
- BOJ12630 Crazy Rows (Large)
- BOJ12776 Swap Space
- BOJ14636 Money for Nothing
- BOJ11001 김치
- BOJ18596 Monster Hunter

구두 수선공

- sum(T_{1..K-1}) * S_K + sum(T_{1..K-1}, T_K) * S_{K+1}
- sum(T_{1..K-1}) * S_{K+1} + sum(T_{1..K-1}, T_{K+1}) * S_K

최적



구두 수선공

- T_K * S_{K+1} <= T_{K+1} * S_K
- T_K / S_K <= T_{K+1} / S_{K+1}
- T_i / S_i 순으로 정렬하면 됨

최적

- Exchange Argument

쿼드 트리

- 열심히 구현하면 된다!

```
int n, a[111][111];
int chk(int x1, int x2, int y1, int y2){
    int ret = a[x1][y1];
    for(int i=x1; i<x2; i++) for(int j=y1; j<y2; j++){
        if(ret != a[i][j]) return -1;
    }
    return ret;
}
void f(int x, int y, int sz){
    int res = chk(x, x+sz, y, y+sz);
    if(res != -1){ cout << res; return; }
    sz /= 2;
    cout << "(";
    f(x, y, sz); f(x, y+sz, sz);
    f(x+sz,y, sz); f(x+sz, y+sz, sz);
    cout << ")";
}
int main(){
    cin >> n;
    for(int i=0; i<n; i++) for(int j=0; j<n; j++){
        char c; cin >> c; a[i][j] = c == '1';
    }
    f(0, 0, n);
}
```

The Shortest does not Mean the Simplest

- 풀이1. 파이썬 - print(pow(a, b, c))

- **이렇게 풀지 마세요**

The Shortest does not Mean the Simplest

- 풀이2. O(log 1e18 * log B)
- A의 B제곱은 O(log B)에 구할 수 있다.

```
ll pw(ll a, ll b, ll mod){
  ll res = 1;
  while(b){
    if(b & 1) res = res * a % mod;
    b >>= 1; a = a * a % mod;
  }
  return res;
}
```

- 10^18 두 개를 곱하면 오버 플로우가 난다.
- 곱셈을 잘 구현해야 한다.

The Shortest does not Mean the Simplest

- 풀이 2-1. __int128_t (128비트 정수형)
- 풀이 2-2. 곱셈 잘 하기

- 1234 = (((1)*10+2)*10+3)*10+4
- x * 1234 = (((1x)*10+2x)*10+3x)*10+4x

```
ll mul(ll a, ll b){
  ll ret = 0;
  string s = f(a);
  for(auto i : s){
    ll now = i - '0';
    ret = ret * 10 % c;
    ret += (now * b) % c;
    ret %= c;
  }
  return ret;
}
```

트리 복구

- preorder와 inorder가 주어졌을 때 postorder 구하는 문제

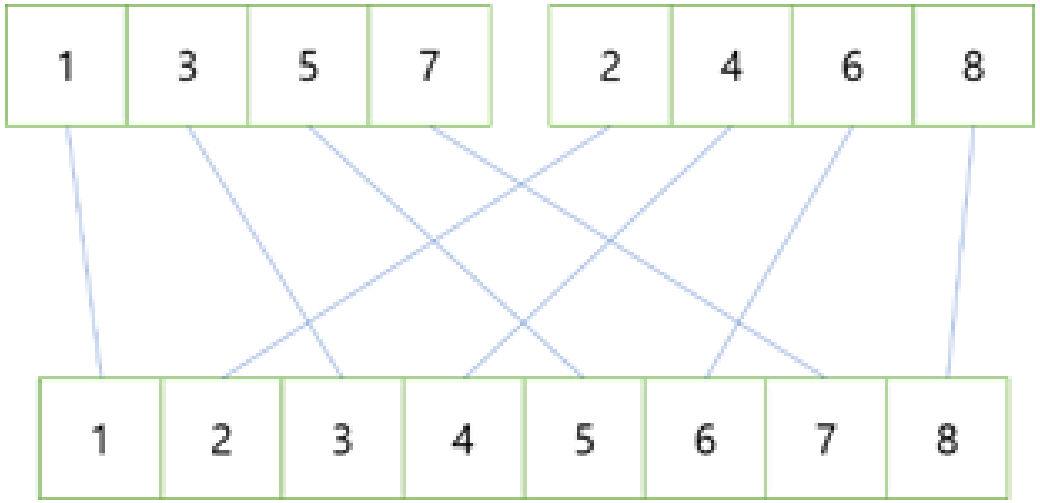
- preorder의 맨 앞에 있는 정점 : 루트
- inorder에서 루트보다 왼쪽에 있는 정점 : 왼쪽 서브 트리
- 루트보다 오른쪽에 있는 정점 : 오른쪽 서브 트리

- 열심히 분할해서 풀자.
- 분할 정복

Counting Inversions

- 세그먼트 트리 / 펜윅 트리 없이 풀자

- Merge Sort!
- merge 단계에서교차점 개수 카운팅



홍수

- 가장 긴 두 개를 이용해 그릇을 만들자
  - N과 (N-1)로 만든 그릇의 최대 용량은 (N-2)(N-1)/2

- N과 (N-1) 사이에 막대기를 적절히 넣어서 K를 만들고
- 안 쓴 막대기들은 바깥쪽에 **그릇이 생성되지 않도록** 잘 배치

캔디캔디

- 가장 큰 것을 1씩 깎아주는 것이 최적
  - 증명은 알아서
  - 이 연산을 20억 번 하는 것이 문제

캔디캔디

- 가장 큰 것을 1씩 깎아주는 것이 최적
  - 증명은 알아서
  - 이 연산을 20억 번 하는 것이 문제

- Parametric Search
  - f(x) : 모든 값을 x 이하로 만들 수 있는가
  - 가능한 최대 x를 찾고, 남은 연산 횟수는 가장 큰 것을 깎아주면 됨
  - 가장 큰 것을 깎는 연산은 최대 N번만 하면 되고, pq로 하면 됨
- O(N log M + N log N)

Passport Control

- 풀이1. 단순 시뮬레이션 O(NK)
- 풀이2. LIS 찾고 제거하는 것을 K번 반복

- 둘 다 쉽다.

Crazy Rows (Large)

- 각 행마다, 1이 나오는 **마지막 위치**가 중요하다.

- 1이 나오는 마지막 위치를 기준으로 버블/삽입 정렬 하면 된다.

Swap Space

- V[i] = B[i] - A[i]라고 하자.

- V[x] >= 0 && V[y] < 0이면 x를 먼저 선택하는 것이 이득
- 둘 다 0 이상이거나 둘 다 0 미만인 경우를 잘 생각해보자.

Swap Space

- V[x] >= 0 && V[y] >= 0인 경우
  - A[x] < A[y]가 되도록 정렬

- 최적인 상태를 -A[x]+B[x]-A[y]+B[y]
- 최적이 아닌 상태를 -A[y]+B[y]-A[x]+B[x]라고 하자.

- 최적일 때의 최저점: min{ -A[x], -A[x]+B[x]-A[y] }
- 최적이 아닐 때의 최저점: min{ -A[y], -A[y]+B[u]-A[x] }
  - -A[y] < -A[x], -A[y] < -A[x]+B[x]-A[y]

Swap Space

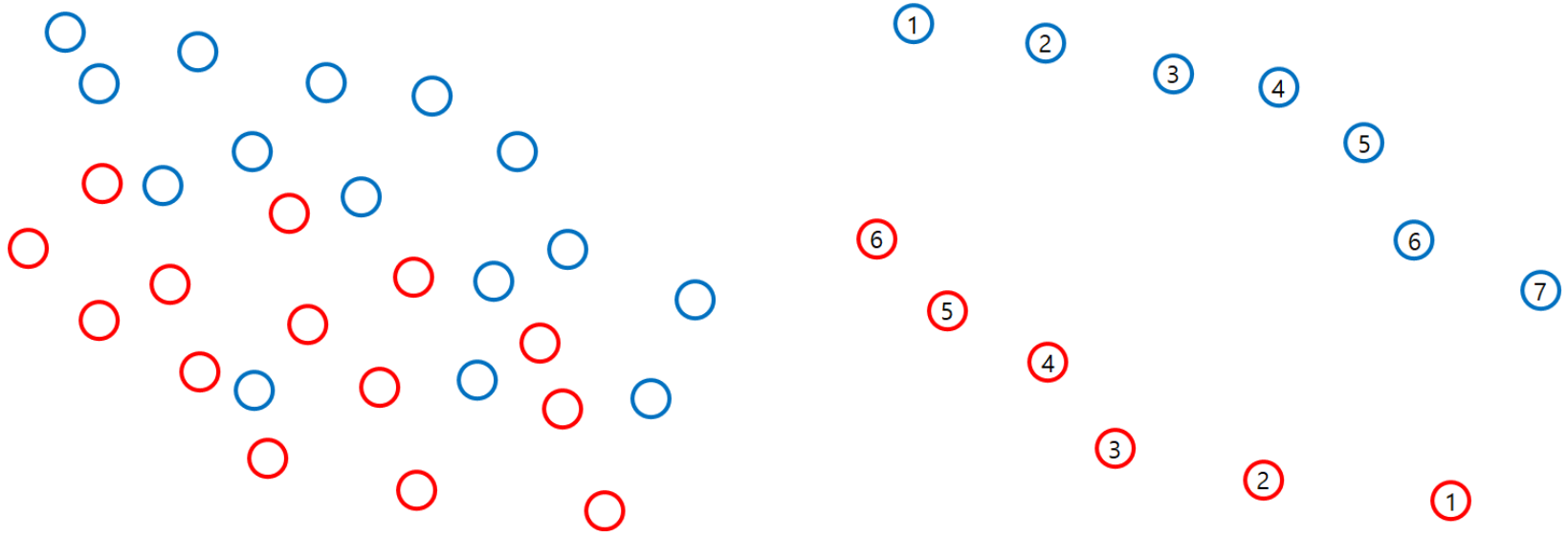
- V[x] < 0 && V[y] < 0인 경우
  - B[x] > B[y]가 되도록 정렬

- 최적인 상태를 -A[x]+B[x]-A[y]+B[y]
- 최적이 아닌 상태를 -A[y]+B[y]-A[x]+B[x]라고 하자.

- 최적일 때의 최저점: min{ -A[x], -A[x]+B[x]-A[y] }
- 최적이 아닐 때의 최저점: min{ -A[y], -A[y]+B[u]-A[x] }
  - -A[x] > -A[y]+B[y]-A[x], -A[x]+B[x]-A[y] > -A[y]+B[u]-A[x]

Money for Nothing

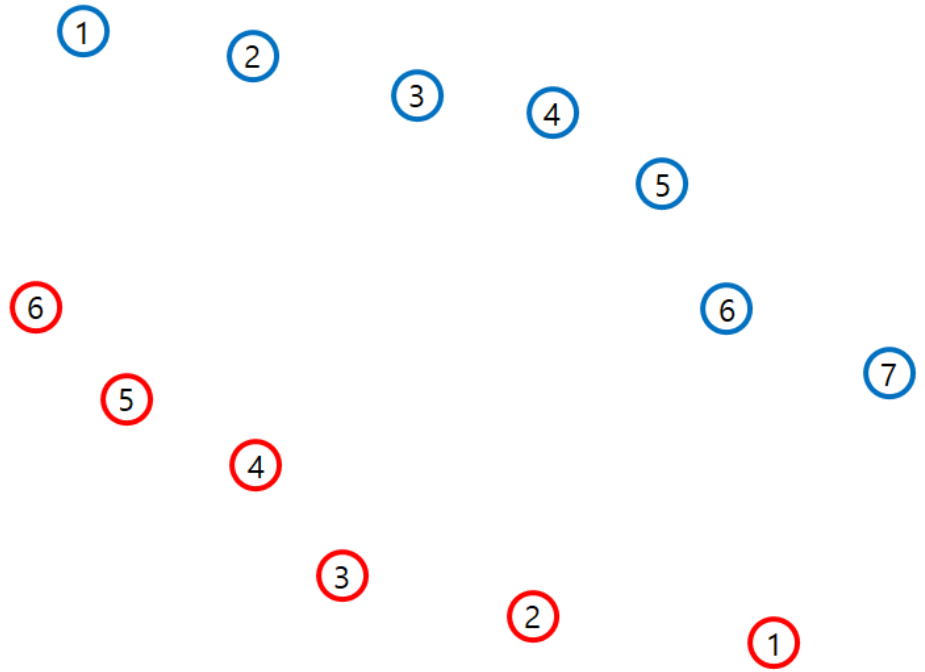
- 임의의 빨간 점 R보다 왼쪽 아래에 빨간 점이 존재한다면?
  - R은 절대 답이 될 수 없다.
- 임의의 파란 점 B보다 오른쪽 위에 파란 점이 존재한다면?
  - B는 절대 답이 될 수 없다.
- 전처리하자.



Money for Nothing

- opt[i] = i번 파란 점을 사용하면서 넓이가 최대가 되는 빨간 점
- opt[i] <= opt[i+1]

- Divide and Conquer Optimization



Money for Nothing

- Divide and Conquer Optimization
  - f(s, e, l, r) : s..e번 파란 점을 사용, 최적이 될 수 있는 빨간 점은 l..r
  - m = (s+e)/2인 m을 잡아서 m에 대한 답을 구하고
  - f(s, m-1, l, opt[m]); f(m+1, e, opt[m], r); 각각 호출

- O(N log M)에 풀림

김치

- 김치를 i번째 날에 꺼낸다고 할 때 김치가 가장 맛있어지는 넣는 날짜를 opt[i]라고 하자.
- 잘 생각해보면 opt[i] <= opt[i+1]이 성립한다.

- Divide and Conquer Optimization

김치 - 증명

- C(i, j) = (j - i) * Tj + Vi -> i날에 넣고 j날에 꺼냄
- a ≤ b ≤ c ≤ d인 a, b, c, d에 대해 아래 부등식이 성립
  - C(a, d) + C(b, c) ≤ C(a, c) + C(b, d) -> Monge Array
- C(i, j) 값으로 이차원 배열을 만들고, 각 행 별로 최댓값의 위치를 보자.
  -> opt[i]
- opt[i]는 무조건 단조 증가한다. 귀류법으로 증명할 수 있다.
- https://justicehui.github.io/hard-algorithm/2020/04/30/monge-array/
  - 3번 성질 참고

Monster Hunter

- Swap Space를 트리에서 한다!

- 아직 방문하지 않은 정점 중 v번 정점을 방문하는 것이 가장 이득이라고 하자.

Monster Hunter

- Swap Space를 트리에서 한다!

- 아직 방문하지 않은 정점 중 v번 정점을 방문하는 것이 가장 이득이라고 하자.
- v의 부모를 방문한 다음에 즉시 v를 방문하는 것이 최적
  - v의 부모와 v를 합쳐줄 수 있다.
- N-1개의 정점으로 구성된 트리에서 또 합치고
  - N-2에서 또 합치고
    - N-1번 합치면 정점 한 개만 남음

Monster Hunter

- 정점을 합치는 것은 Union Find, 최적인 정점을 뽑는 것은 pq
- O(N log N)에 가능

- 정점을 합치는 것은 (A[x], B[x]), (A[y], B[y])가 있을 때

mn = min{-A[x], -A[x]+B[x]-A[y]}라고 하면

(mn, -A[x]+B[x]-A[y]+B[y] - mn)으로 수정하면 됨