Day9 휴리스틱

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

문제 목록

- BOJ18192 보고 정렬
- BOJ2912 백설공주와 난쟁이
- BOJ2582 동전 뒤집기 2
- BOJ8873 미술 시간
- BOJ2389 세상의 중심에서...
- CMS 랜덤 셔플
- CMS 2-tsp

보고 정렬

- 랜덤의 성질: 모든 경우의 수를 동일한 확률로 발생시킴

- 길이가 N인 배열에서 원하는 수 하나를 원하는 위치로 옮기기 위해 섞어야 하는 횟수의 기댓값 : O(N)
  - (N-1)! / N! = 1/N

보고 정렬

- 퀵정렬을 수행하자.
  - pivot 하나 잡고, A[pivot] = pivot이 될 때까지 shuffle_array 수행
  - [0, pivot-1]과 [pivot+1, N-1] 구간에 대해 재귀적으로 수행

백설공주와 난쟁이

- 틀릴 확률이 p인 알고리즘 A가 있다고 하자.
  - 맞을 확률은 (1 - p)이다.
- A를 1번 수행하면 틀릴 확률이 p이지만, 100번 수행하면 p^100이다.
  - 100번의 수행 중 한 번이라도 정답을 내면 된다.
- 0 <= p < 1이므로 여러 번 수행할수록 틀릴 확률이 감소한다.

백설공주와 난쟁이

- 쿼리로 들어온 구간 [s, e]의 절반 이상이 같은 수 x로 구성되어 있다고 하자.
- 랜덤으로 원소를 선택하면 x를 선택할 확률이 ½ 이상이다.
  - 틀릴 확률은 ½ 이하이다.

백설공주와 난쟁이

- 쿼리로 들어온 구간 [s, e]의 절반 이상이 같은 수 x로 구성되어 있다고 하자.
- 랜덤으로 원소를 선택하면 x를 선택할 확률이 ½ 이상이다.
  - 틀릴 확률은 ½ 이하이다.

- 틀릴 확률이 ½인 알고리즘을 100번 반복해주면 틀릴 확률이 (½)^100이 된다.

백설공주와 난쟁이

- 어떤 수가 임의의 구간의 절반 이상을 차지하는지 확인
  - 각 수마다 등장하는 인덱스를 정렬된 상태로 저장
  - upper_bound와 lower_bound를 이용해 확인

백설공주와 난쟁이

- Mo’s Algorithm + Segment Tree 쓰는 풀이도 있긴 하다.

동전 뒤집기 2

- Simulated Annealing
  - https://koosaga.com/3 를 참고하자.
    - 2~3년 전에 [redacted] 면접에서 SA를 물어봤다는 소문이 있던데...

미술 시간

- 다양한 풀이가 존재한다. 가장 간단한 풀이를 소개한다.

미술 시간

- 그림들을 잘 보자.

- 스타일 4(색면회화) 그림은 큰 덩어리 2~4개로 이루어져 있다.
- 스타일 1(신조형주의) 그림은 큰 덩어리 XX개로 이루어져 있다.
- 스타일 2(인상주의) 그림은 초록색이 비교적 많다.
- 스타일 3(표현주의) 그림은 무작위로 뿌려진 느낌이 강하다.

- 덩어리의 개수를 잘 카운팅하자.

미술 시간

- 사실 덩어리 개수보다는 (인접한 픽셀의 색깔 차이)의 합을 보는게 정확도가 높다.

- 적절한 상수 a, b, c를 잡아서
  - sum/hw가 a 미만이면 4
  - b 미만이면 1
  - c 미만이면 2
  - c 이상이면 3으로 결정하면 된다.

```
for(int i=1; i<h; i++){
    for(int j=1; j<w; j++){
        sum += abs(r[i+1][j] - r[i][j]);
        sum += abs(r[i][j+1] - r[i][j]);
        sum += abs(g[i+1][j] - g[i][j]);
        sum += abs(g[i][j+1] - g[i][j]);
        sum += abs(b[i+1][j] - b[i][j]);
        sum += abs(b[i][j+1] - b[i][j]);
    }
}
```

세상의 중심에서

- 경사 하강법 쓰면 된다.
- 거리 함수는 Unimodal하고, Unimodal 함수의 합/최대/최소도 Unimodal하기 때문에 경사 하강법으로 답을 찾을 수 있다.


```
double alpha = 0.1;
double r;
for(int i=0; i<30303; i++){
    int idx = 1;
    r = 0;
    for(int j=1; j<=n; j++){
        double now = hypot(xx - x[j], yy - y[j]);
        if(now > r) r = now, idx = j;
    }
    xx += (x[idx] - xx) * alpha;
    yy += (y[idx] - yy) * alpha;
    alpha *= 0.999;
}
```

랜덤 셔플

- 1x년 전 Google Code Jam 기출

- method_1은 가능한 N!가지 경우의 수를 동일한 확률로 생성한다. -> 귀납법으로 증명 가능
- method_2는 method_1에 비해 작은 수가 뒤로 갈 확률이 높다.

랜덤 셔플

- method_1은 가능한 N!가지 경우의 수를 동일한 확률로 생성한다. -> 귀납법으로 증명 가능
- method_2는 method_1에 비해 작은 수가 뒤로 갈 확률이 높다.

- A[i] >= i인 쌍의 개수에 따라 적절히 판단해주면 된다.
- 판단하는 기준 값은 로컬에서 직접 돌려서 규칙을 찾으면 된다.

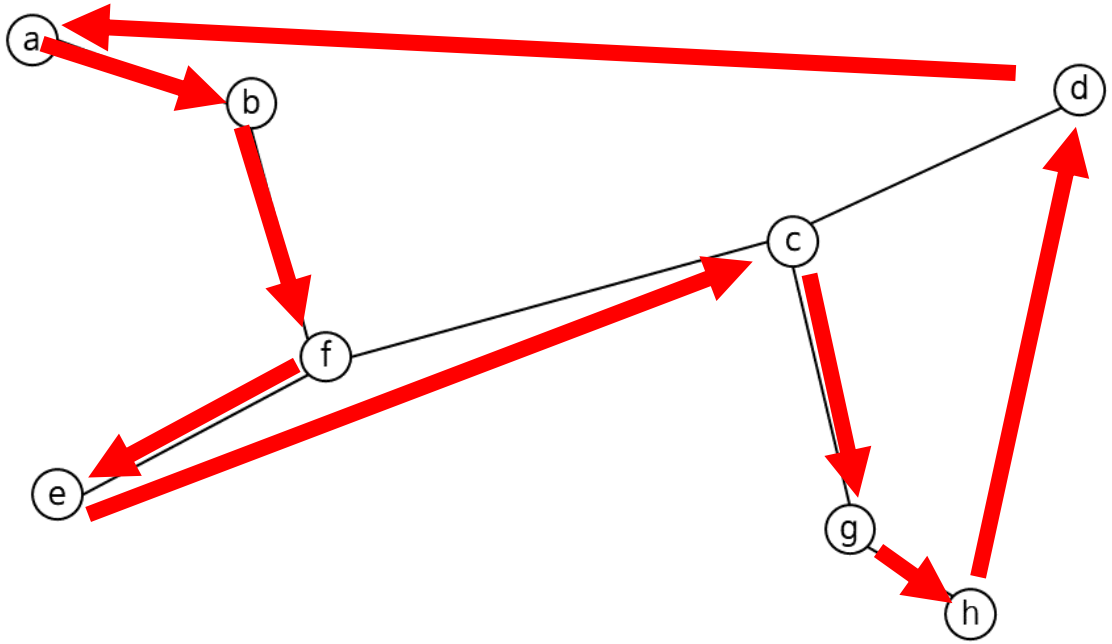
2-tsp

- tsp는 O(N^2 * 2^N)보다 빠르게 풀기 어렵다.
- 유클리드 거리 TSP의 2배 이하 근사 알고리즘은 O(N^2 log N) 정도에 쉽게 구할 수 있다.
  - 2-approximation TSP

2-tsp

- 2-tsp를 푸는 휴리스틱 알고리즘
  - MST를 구한다.
  - MST에서 dfs ordering을 매긴다.
  - dfs ordering 순서대로 방문한다.
  - 시작점으로 돌아온다.

2-tsp



2-tsp

- Theorem. 이 휴리스틱의 근사도는 2이다.
- proof.
  - TSP의 최적해에서 간선 하나를 제거하면 Spanning Tree가 나온다.
    - length(MST) <= length(TSP - {e}) <= length(TSP)
  - MST의 각 간선을 두 번 지나면 모든 정점을 지나서 시작점으로 돌아오는 경로(euler tour) C를 얻을 수 있다. C의 길이는 MST 길이의 2배 이하이다.
    - length(C) = 2length(MST) <= 2length(TSP)
  - 정점 u에서 정점 v로 갈 때 다른 정점을 거치지 않고 직선으로 가는 경로 result는 C보다 길이가 짧다.
    - length(result) <= length(C) <= 2length(TSP)