2021 IUPC

Inha University Programming Contest

인하대학교 컴퓨터공학과
Department of Computer Engineering

NAVER D2

STARTLINK

A. 스키테일 암호

First solved : 수퍼겁쟁이들의 쉼터 (1 min)
맞은 팀 수 : 22
정답률 : 91.667%
출제자 : suun

2021 IUPC
Inha University Programming Contest

A. 스키테일 암호

- 1번 문자, $K+1$번 문자, $2K+1$번 문자… 를 출력하면 됩니다.
- 반복문으로 구현할 수 있습니다.
- 시간 복잡도: $O(|S|)$

B. 오델로

First solved : 런타임 테러 (19 min)
맞은 팀 수 : 14
정답률 : 22.951%
출제자 : jjmjm2002

2021 IUPC
Inha University Programming Contest

B. 오델로

- 모든 빈 공간에 흑돌을 한 번씩 놓아봐도 됩니다.
- 가로, 세로, 대각선 총 8방향으로 총 몇 개의 돌을 뒤집을 수 있는지 확인합니다.

- 뒤집어지는 백돌이 한 개도 없다면 놓을 수 없는 자리입니다.
- 놓을 수 있는 자리 중 백돌을 가장 많이 뒤집는 위치를 완전 탐색으로 찾으면 됩니다.

- 시간 복잡도: $O(N^3)$

C. 균형 삼진법

First solved : 수퍼겁쟁이들의 쉼터 (34 min)
맞은 팀 수 : 14
정답률 : 28.571%
출제자 : wjdclgns12

2021 IUPC
Inha University Programming Contest

C. 균형 삼진법

- 덧셈과 진법에 대한 이해를 묻는 문제로, 푸는 방법이 정말 다양합니다.
- 어떤 수 x를 k자리 균형 삼진법 수로 나타낸다면,
- x의 범위는 $-\left\lfloor \frac{3^k}{2} \right\rfloor \leq x \leq \left\lfloor \frac{3^k}{2} \right\rfloor$ 입니다.
- 이제 주어진 수를 균형 삼진법으로 변환했을 때 몇 자리 수가 되는지 알 수 있습니다.

C. 균형 삼진법

- x를 균형 삼진법으로 나타냈을 때의 결과가 k자리라고 가정했을 때, 가장 높은 자릿수는 다음의 식으로 정해집니다.

- $-\left\lfloor \frac{3^k}{2} \right\rfloor \le x < -\left\lfloor \frac{3^{k-1}}{2} \right\rfloor \Rightarrow -1$
- $-\left\lfloor \frac{3^{k-1}}{2} \right\rfloor \le x \le \left\lfloor \frac{3^{k-1}}{2} \right\rfloor \Rightarrow 0$
- $\left\lfloor \frac{3^{k-1}}{2} \right\rfloor < x \le \left\lfloor \frac{3^k}{2} \right\rfloor \Rightarrow 1$

- x에서 (가장 높은 자릿수)$\times 3^k$를 빼고 남은 수 x'에 대해서 재귀적으로 자릿수를 결정해주면 됩니다.

2021 IUPC
Inha University Programming Contest

C. 균형 삼진법

- 다른 풀이는 3진법 표현을 구한 뒤, 변화를 주는 방법입니다.

- 3진법의 $0 \times 3^{k+1} + 2 \times 3^{k}$ 은,
- 균형 삼진법의 $1 \times 3^{k+1} + (-1) \times 3^{k}$ 과 같습니다.

- 즉, 3진법 표현에서 2를 -1로 변경하고 carry를 올리면 됩니다.

- 음수는 절대값의 3진법을 구해서 1을 T로, T를 1로 바꿔주는 것으로 구할 수 있습니다.

C. 균형 삼진법

- 예외 사항)
  - ✓ `0`이 입력으로 주어지는 경우 예외처리를 해주어야 합니다.
  - ✓ 결과가 `0`으로 시작하지 않아야 합니다.

- 시간 복잡도: $O(\log_3 N)$

K. 꿀벌 승연이

First solved : 수퍼겁쟁이들의 쉼터 (153 min)
맞은 팀 수 : 5
정답률 : 19.231%
출제자 : ruz

2021 IUPC
Inha University Programming Contest

K. 꿀벌 승연이

- 승연이가 이동할 수 있는 경로를 그래프로 나타내 보면 DAG(Directed Acyclic Graph)가 됩니다.
- DAG는 사이클이 없어서 참조 투명성을 잃지 않는 특징이 있습니다.
- 즉, 다이나믹 프로그래밍으로 문제를 해결할 수 있습니다.

K. 꿀벌 승연이

- $DP[y][x]$를 $(1, 1)$칸에서 (y, x)칸으로 가는 경우의 수라고 정의하면, 다음과 같은 점화식을 얻을 수 있습니다.

- if x is even,
  $DP[y][x] = DP[y-1][x] + DP[y][x-1] + DP[y+1][x-1]$,

- if x is odd,
  $DP[y][x] = DP[y-1][x] + DP[y-1][x-1] + DP[y][x-1]$

- (y, x)가 구멍 칸인 경우, $DP[y][x] = 0$ 이고,

- $(1, 1)$칸은 $DP[1][1] = 1$ 입니다.

2021 IUPC
Inha University Programming Contest

K. 꿀벌 승연이

- DP 정의에 따라 다른 점화식으로도 풀 수 있습니다.
- 슬라이딩 윈도우 메모리 최적화가 가능합니다. 항상 2개 열만 필요하게 됨을 이용합니다.
- 시간 복잡도는 $O(NM)$ 입니다.

F. IUPC와 비밀번호

First solved : MuYaHo (125 min)
맞은 팀 수 : 3
정답률 : 4.545%
출제자 : wjdclgns12

2021 IUPC
Inha University Programming Contest

F. IUPC와 비밀번호

- 각 암호 후보 T_i에서 길이가 $|S|$인 어떤 부분 문자열 T_i'를 생각해봅시다.

- T_i'의 알파벳의 구성과 개수가 S와 완전히 같다면, T_i'은 S를 랜덤하게 뒤섞은 것으로 볼 수 있습니다.

- T_i'의 앞 뒤로 임의의 문자열을 붙이고, 임의의 위치의 문자 1개를 골라 다른 문자로 바꾸는 경우는 두 가지가 있습니다.

  a. T_i'에 포함되는 문자를 바꾼 경우
     - 문자 하나는 1개 많아지고, 하나는 1개 적어집니다.

  b. 앞 뒤로 붙여진 임의의 문자열 중에서 하나를 바꾼 경우
     - T_i'과 S의 알파벳 구성과 개수가 달라지지 않습니다.

2021 IUPC
Inha University Programming Contest

F. IUPC와 비밀번호

- 따라서, T_i'와 S의 알파벳 구성과 개수를 비교하는데,

1. 정확히 2개의 알파벳이 한 쪽은 1개 많고 다른 쪽은 1개 적으며 나머지 구성은 같은 경우
2. 모든 알파벳 구성이 완전히 같은 경우

- 둘 중 하나라도 존재한다면, T_i는 비밀번호였을 가능성이 있습니다.

2021 IUPC
Inha University Programming Contest

F. IUPC와 비밀번호

- 단, $|T_i| == |S|$ 인 경우, T_i'에 포함되지 않는 문자를 바꾸는 것은 불가능하므로, 알파벳 구성과 개수가 완전히 같으면 비밀번호일 수 없습니다.
- $|T_i| < |S|$ 인 경우, 만들 수 있는 방법이 없으므로 불가능합니다.
- 길이가 $|S|$인 모든 부분 문자열에 해당하는 알파벳 개수는 슬라이딩 윈도우를 이용하여 빠르게 계산할 수 있습니다.
- 시간 복잡도: $O(|S| + \sum |T_i|)$

2021 IUPC
Inha University Programming Contest

D. aging

First solved : 수퍼겁쟁이들의 쉼터 (101 min)
맞은 팀 수 : 2
정답률 : 11.111%
출제자 : ruz

2021 IUPC
Inha University Programming Contest

D. aging

- T초에 아직 실행되지 않고 우선순위 큐에 남아있는 i번 프로세스의 우선순위는 $p_i + T - t_i$ 입니다.
- 구하는 값은 $max_{i \in PQ}(p_i + T - t_i)$인데,
- 여기서 T는 i와 관계 없는 **상수**이므로 $max_{i \in PQ}(p_i - t_i)$로 정리할 수 있습니다.
- 즉, T초는 프로세스 비교에 영향을 주지 않습니다.

D. aging

- i, j번째 프로세스를 비교할 때,

- $p_i - t_i$와 $p_j - t_j$가 다르다면 둘 중 큰 쪽,
- b_i와 b_j가 다르다면 둘 중 작은 쪽,
- 모두 같다면 i와 j중에 작은 쪽이 먼저 처리되도록 하면 됩니다.

- Heap 자료구조를 이용할 수 있습니다.

- 시간 복잡도: $O(N \log N)$

I. 판치기

First solved : 수퍼겁쟁이들의 쉼터 (81 min)
맞은 팀 수 : 2
정답률 : 5.405%
출제자 : wjdclgns12

2021 IUPC
Inha University Programming Contest

I. 판치기

- 앞 면이 보이는 동전의 개수가 같다면, 같은 상태라고 생각할 수 있습니다.
- 한 번의 K-뒤집기로 이동할 수 있는 두 상태를 간선으로 이어주면 그래프를 정의할 수 있습니다.
- 이 그래프의 시작/도착점이 주어졌을 때, 최단 거리를 구하는 문제가 됩니다.
- 모든 간선의 가중치가 같으므로, 간단한 BFS로 답을 구할 수 있습니다.
- 시간 복잡도: $O(N^2)$

E. 두 반으로 나누기

First solved :	Not Solved
맞은 팀 수 :	0
정답률 :	0.0%
출제자 :	jjmjm2002

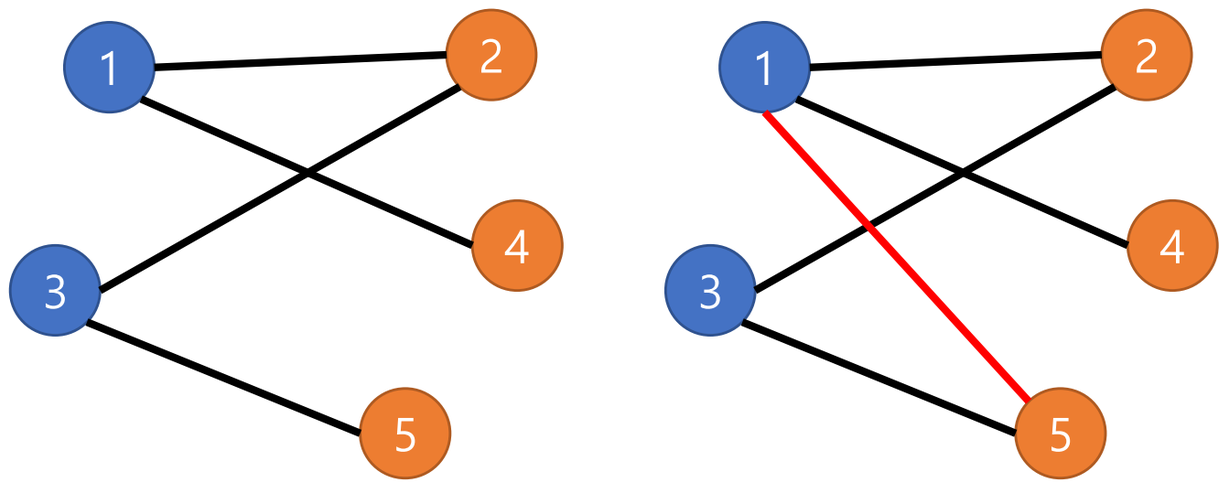
2021 IUPC
Inha University Programming Contest

E. 두 반으로 나누기

- 간선을 순서대로 제거하면서 그래프가 이분그래프가 되는지 확인하는 문제입니다.
- 이분그래프 여부는 flood fill을 응용하여 확인할 수 있습니다.
- 하지만, 간선을 하나 지울 때마다 매번 이분그래프 여부를 확인하는 것은 너무 느린 풀이입니다.
- 리스트의 모든 간선을 지우더라도 그래프는 여전히 연결그래프 임에 주목해봅시다.

E. 두 반으로 나누기

- 모든 간선을 지운 연결그래프가 이분그래프라면,



- 서로 다른 색으로 칠해진 두 정점을 간선으로 이을 때는 여전히 이분그래프가 됩니다.

E. 두 반으로 나누기

- 모든 간선을 지운 상태에서 시작하여, 간선을 역순으로 추가하면서 어느 순간에 이분그래프가 아니게 되는지 보면 됩니다.

- 초기 그래프가 이분그래프 인지 검사하고,
- K개 간선을 추가할 때마다, $O(1)$에 이분그래프 여부를 판단하면 문제를 해결할 수 있습니다.

- 시간 복잡도: $O(N + M + K)$

E. 두 반으로 나누기

- 이분 탐색으로 해결하는 방법도 있습니다.
- 이분그래프 판정 결과가 true→false 로 단조 감소하므로 이분 탐색을 통해 결과가 바뀌는 시점을 찾을 수 있습니다.
- x개 만큼 간선을 남겼을 때, 이분그래프 여부를 판정해보고, 이분그래프라면 x를 증가, 이분그래프가 아니라면 x를 감소시켜보면 됩니다.
- $\log K$번의 이분그래프 판정을 시도해보면 되므로, 시간 복잡도는 $O((N+M)\log K)$ 입니다.

2021 IUPC
Inha University Programming Contest

G. 최단최단경로

First solved :	Not Solved
맞은 팀 수 :	0
정답률 :	0.0%
출제자 :	suun

2021 IUPC
Inha University Programming Contest

G. 최단최단경로

- 우선 최단경로를 구해야 합니다. 최단경로는 Dijkstra 알고리즘으로 구할 수 있습니다.
- **최단최단 경로**를 구하기 위해서는 최단거리로 이동하는데 쓰이는 간선만을 이용해야 합니다.
- 간선 $(u,\ v,\ w)$에 대해서 $dist(1,u)+w+dist(v,n)=dist(1,n)$인 경우 해당 간선은 유효한 간선이 됩니다.
- 유효한 간선만으로 그래프를 재구성하면 DAG로 나타나게 됩니다.

2021 IUPC
Inha University Programming Contest

G. 최단최단경로

- 이제 다음과 같은 DP 점화식을 이용하면 **최단최단 경로**로 이동했을 때 몇 개의 간선을 거치는지 알 수 있습니다.

- $DP[v] = \min(DP[v], DP[u] + 1)$

- 그러면 또다시, **최단최단 경로**로 이동하는데 쓰이는 간선만을 이용해야 된다는 사실을 알 수 있습니다.

G. 최단최단경로

- 서로 다른 **최단최단 경로**가 몇가지 존재하는지 다음과 같은 점화식을 이용해 알 수 있습니다.
- $DP_2[v] = (DP_2[v] + DP_2[u]) \% MOD$
- +각 정점까지의 거리를 {이동 거리, 사용한 간선 수}로 정의하면, Dijkstra 한 번으로 **최단최단 경로**를 알아낼 수도 있습니다.
- 시간 복잡도: $O((N+M)\log N)$

J. 사탕나무

First solved :	Not Solved
맞은 팀 수 :	0
정답률 :	0.0%
출제자 :	wjdclgns12

2021 IUPC
Inha University Programming Contest

J. 사탕나무

- 모든 정점을 한 번씩 기준으로 하여 거리가 K인 점을 매번 구해주는 방법은 너무 느립니다.
- 일단, 임의의 정점을 트리의 루트로 정의하고 각 정점을 루트로 하는 서브 트리에 대해,
  거리가 $0 \sim K$인 정점의 개수를 구해 놓읍시다.
- 이 작업은 간단하게 트리 다이나믹 프로그래밍으로 할 수 있습니다.

J. 사탕나무

- 현재 u번 정점이 루트라고 합시다.
- u와 인접한 v번 정점을 루트로 바꾸려고 한다면, v를 u에서 떼어 낸 다음 반대로 u를 v의 자식으로 붙이면 됩니다.
- 즉, 다음과 같이 $O(K)$개 값만 수정하면 됩니다.
- for i in range(K) :

  $DP[u][i+1]$ $-=$ $DP[u][i]$

  $DP[v][i+1]$ $+=$ $DP[u][i]$

J. 사탕나무

- 그러면, 그래프를 DFS로 순회하면서 모든 정점이 한 번씩 루트가 되도록 DP 테이블을 수정하면서 답을 찾을 수 있습니다.
- 이렇게 트리의 루트를 바꾸었다고 생각하고 DP 테이블을 수정하는 테크닉을 re-rooting이라고 합니다.
- 시간 복잡도: $O(NK)$

H. 난민

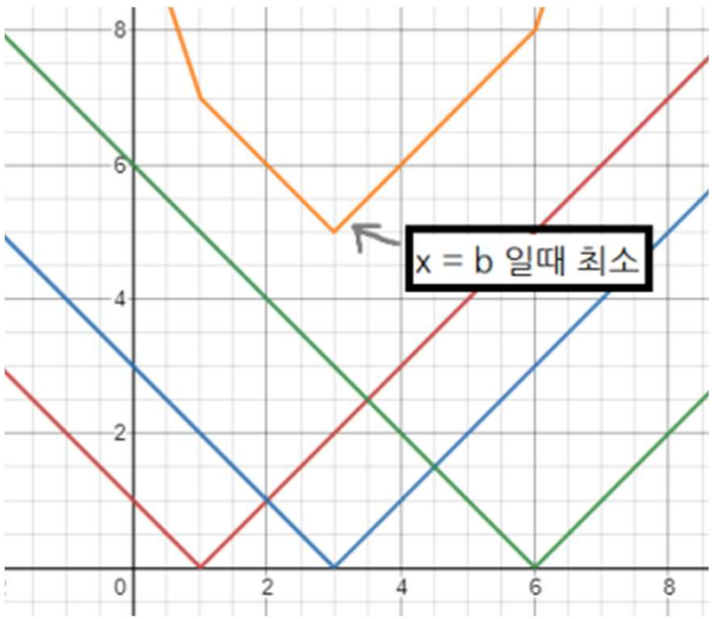
First solved :	Not Solved
맞은 팀 수 :	0
정답률 :	0.0%
출제자 :	wjdclgns12

2021 IUPC
Inha University Programming Contest

H. 난민

- 난민의 x좌표는 답에 영향을 주지 않습니다. 단순히 x좌표의 절대값만 마지막에 더해주면 됩니다.
- 그러니 난민의 y좌표만 고려해봅시다. 난민들의 y좌표가 주어졌을 때, 정수시설을 어디에 설치해야 최적일까요?
- 아래와 같은 식이 최소가 되는 Y값임을 알 수 있습니다.
- $|Y - y_1| + |Y - y_2| + \ ... + |Y - y_n|$

H. 난민



- 그래프로 그려보면 다음과 같습니다.
- $y = |x - a|$
- $y = |x - b|$
- $y = |x - c|$
- $y = |x - a| + |x - b| + |x - c|$

- 이와 같이 난민들의 y좌표 중 중앙값이 답이 된다는 사실을 알 수 있습니다.

- 따라서, 난민들의 y좌표가 추가됨에 따라 중앙값을 계속해서 갱신하는 문제가 됩니다.

H. 난민

- 동적 자료의 중앙값을 구하는 문제는 두 개의 Heap을 이용한 풀이가 잘 알려져 있습니다.
- 또는, 좌표 압축 후 구간 쿼리 자료구조를 이용해서 계산하거나, Set 등의 균형 이진탐색 트리를 이용하는 방법도 있습니다.
- 시간 복잡도: $O(N \log N)$

L. calculate! 3

First solved :	Not Solved
맞은 팀 수 :	0
정답률 :	0.0%
출제자 :	ruz

2021 IUPC
Inha University Programming Contest

L. calculate! 3

- Rooted tree에서 $u \to v$경로 상의 가중치를 모두 XOR 한 값을 **경로의 XOR** 이라고 하고, (uv)로 나타낸다고 합시다.
- Root 정점이 r이라면, 임의의 두 정점 u, v에 대해 경로의 XOR값은 $(uv) = (ru) \oplus (rv)$ 와 같습니다.
- 문제에서 가중치는 1~30 까지 수로 주어지는데, XOR 연산을 하다 보면 0이나 31도 경로의 XOR 값이 될 수 있다는 점에 주의해야 합니다.

2021 IUPC
Inha University Programming Contest

L. calculate! 3

- 이를 통해, 경로의 XOR 값이 c인 경로의 개수는,

$$\sum_{i=0}^{31} n(\{x\,|(rx) = i\}) \times n(\{x|(rx) = (i \oplus c)\})$$

임을 알 수 있습니다. $n(A)$은 집합 A의 원소의 개수입니다.

- 그럼, 간선의 가중치가 바뀌는 경우는 어떻게 처리할 수 있을까요?

2021 IUPC
Inha University Programming Contest

L. calculate! 3

- $\text{depth}(a) < \text{depth}(b)$ 일 때, 간선 $a \to b$의 가중치가 c에서 c'으로 바뀌면,
- b 정점을 루트로 하는 서브 트리의 모든 정점 x에 대해 (rx)가 $(rx) \oplus (c \oplus c')$ 으로 변화합니다.
- b 정점을 루트로 하는 서브 트리에 32가지 가중치가 각각 몇 개씩 있는지 관리하고,
- i 가중치 개수와 $i \oplus (c \oplus c')$ 가중치 개수가 서로 바뀌도록 하면 됩니다.

L. calculate! 3

- 이를 서브 트리의 모든 정점에 대해 처리해주어야 하는데, 모든 정점을 돌아보며 하나씩 수정을 하면 너무 느립니다.
- Euler Tour Technique을 이용하면 트리를 하나의 선형 배열로, 각 서브 트리들은 연속한 부분 배열로 볼 수 있습니다. 이제 이 배열을 구간 트리로 관리할 수 있습니다.
- 구간을 갱신해야 하므로, lazy propagation이 필요합니다.
- 시간 복잡도: $O(32 \times (N+M)\log N)$