

SUAPC 2021s 풀이

Official Solutions

신촌지역 대학생 프로그래밍 동아리 연합

2021년 8월 28일

문제	의도한 난이도	출제자
A 휴먼 파이프라인	Easy	전해성 ^{seastar105}
B 선인장의 독립집합	Challenging	이국렬 ^{lky7674}
C 조각 체스판	Medium	박수현 ^{shiftpsh}
D 반짝반짝 2	Easy	이국렬 ^{lky7674}
E 문자열 조작의 달인	Hard	김규진 ^{snrnsidy}
F Flat Earth	Easy	전해성 ^{seastar105}
G 기지국 업그레이드	Hard	이혜아 ^{ho94949}
H 재활용 캠페인	Medium	이혜아 ^{ho94949}
I 브런치북	Hard	이혜아 ^{ho94949}
J 사이클	Hard	이국렬 ^{lky7674}
K 수요응답형 버스	Medium	김규진 ^{snrnsidy}
L 다꾸	Challenging	박수현 ^{shiftpsh}

A. 휴먼 파이프라인

sort

출제진 의도 - **Easy**

- ✓ 제출 257번, 정답 34팀 (정답률 13.229%)
- ✓ 처음 푼 팀: 생수@홍익대 (xkd1aldfjtn1, hwon233, dicohy27), 9분
- ✓ 출제자: 전해성^{seastar105}

A. 휴먼 파이프라인

- ✓ 모든 사람이 두 팀중 하나엔 들어가야 하기 때문에 제일 작업속도가 느린 사람이 속도가 속한 팀의 최소 속도는 제일 작은 v_i 가 됩니다.
- ✓ 주어진 작업속도 v_i 를 오름차순으로 정렬한 것을 a_1, a_2, \dots, a_N 라고 합시다.
- ✓ 이러면 한 팀의 최소 속도는 a_1 이 됩니다.

A. 휴먼 파이프라인

- ✓ 모든 상자를 빠르게 옮기기 위해서는 당연히 각 팀의 작업속도가 빨라야 할 것입니다.
- ✓ 팀의 작업속도는 속한 팀원의 작업 속도 중의 최소 작업속도와 팀원의 수에 의해 결정되는데 한 팀의 최소 속도는 무조건 a_1 입니다.
- ✓ a_1 이 속한 팀의 인원 수를 k 라고 합시다. 다른 팀의 최소 작업속도를 가장 크게 만들기 위해선 a_1 이 속한 팀은 a_1, a_2, \dots, a_k 로 이뤄져야 합니다.

A. 휴먼 파이프라인

- ✓ 이제 a_1 이 속한 팀의 인원수가 k 일 때, 두 팀의 속도를 결정할 수 있습니다. 이 속도를 각각 V_1, V_2 라고 합시다.
- ✓ 상자를 각 팀에게 잘 분배하여 빠르게 끝나는 경우를 구해야 합니다.
- ✓ 상자를 각 팀에 분배한다고 생각하지 말고 두 팀이 같이 작업한다고 생각하면 분배를 고려할 필요가 없어집니다. 두 팀의 속도가 V_1, V_2 일 때 작업속도는 아래와 같습니다.

$$\left[\frac{K}{V_1 + V_2} \right]$$

A. 휴먼 파이프라인

- ✓ 따라서, a_1 이 속한 팀의 인원수를 1명, 2명, \dots , $N - 1$ 명까지 모든 경우를 시도해보고 그 중에 최솟값이 답이 됩니다.
- ✓ 정렬을 하는 데 $\mathcal{O}(N \log N)$ 이 걸리고 최솟값을 구하는 데 $\mathcal{O}(N)$ 입니다.

B. 선인장의 독립집합

cactus, bcc, dp

출제진 의도 - **Challenge**

- ✓ 제출 76번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: — (—), —분
- ✓ 출제자: 이국렬^{1ky7674}

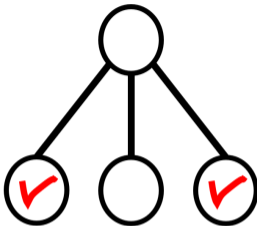
B. 선인장의 독립집합

- ✓ 원래 Camp Contest에 BCC 관련 문제로 출제할 예정이었으나 난이도 문제로 SUAPC로 옮겨진 문제입니다.
- ✓ Cactus Graph에서 바로 생각하는 건 어렵습니다.
- ✓ 때문에 우선 더 쉬운 문제인 트리의 독립집합 문제를 보도록 합시다.

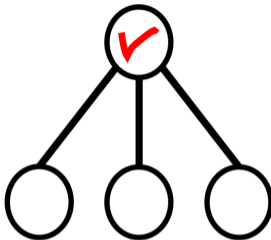
B. 선인장의 독립집합

- ✓ 트리의 독립집합 문제는 다음과 같이 dp 식을 정의하면 선형 시간에 구할 수 있습니다.
 - $dp(i, 0)$: i 번째 노드가 루트인 서브 트리에서 i 번째 노드를 선택하지 않았을 때 최대 독립 집합의 크기
 - $dp(i, 1)$: i 번째 노드가 루트인 서브 트리에서 i 번째 노드를 선택했을 때 최대 독립 집합의 크기

B. 선인장의 독립집합

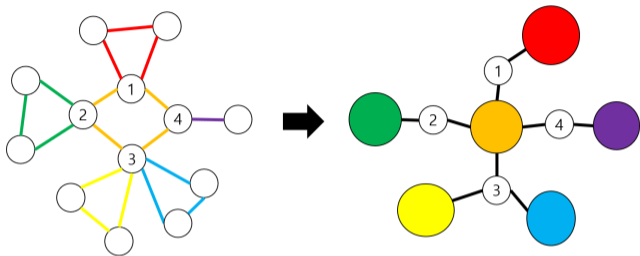


- ✓ $dp(i, 0)$: i 번째 노드가 루트인 트리에서 i 번째 노드를 선택하지 않았을 때 최대 독립 집합의 크기
- ✓ 자식 노드를 선택 할 수도 있고, 안 할수도 있습니다.
- ✓ $dp(i, 0) = \sum_{j \in child(i)} \max(dp(j, 0), dp(j, 1)).$



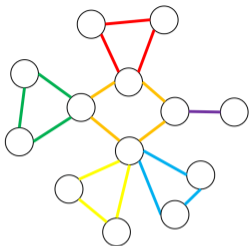
- ✓ $dp(i, 1)$: i 번째 노드가 루트인 트리에서 i 번째 노드를 선택했을 때 최대 독립 집합의 크기
- ✓ 자식 노드는 무조건 선택하면 안 됩니다.
- ✓ $dp(i, 1) = 1 + \sum_{j \in \text{child}(i)} dp(j, 0)$.

B. 선인장의 독립집합



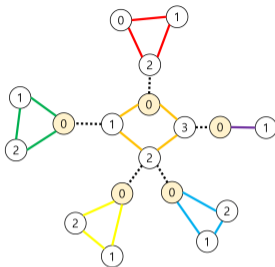
- ✓ 선인장 그래프에서 최대 독립집합의 크기를 구하는 것과 트리에서 구하는 것과 어떤 관계?
- ✓ 사이클 단위로 묶으면 트리가 만들어집니다.
- ✓ 트리의 독립집합 문제처럼 부모 사이클부터 탐색하면 됩니다.

B. 선인장의 독립집합



- ✓ 우선 그래프를 cycle 단위로 나눠줍니다.
- ✓ 각 정점별로 몇 번째 cycle과 연결되어 있는지를 인접 리스트로 저장합니다.
- ✓ 이 과정은 단절점을 찾고, 단절점을 기준으로 BCC로 나누면 선형시간에 할 수 있습니다.
- ✓ 이후 풀이에서 편의상 간선 1개로 구성된 BCC도 2개의 정점으로 이루어진 cycle로 취급할 것입니다.

B. 선인장의 독립집합



- ✓ 각 cycle 별로 속한 정점들의 번호를 순서대로 매겨 줍니다.
- ✓ 부모 cycle과 연결된 정점을 0번째로 합시다.
- ✓ 이 과정은 dp식을 간단하게 만들어 줍니다.
- ✓ BCC 만드는 과정 중에서 번호 순서대로 방문하기에 이를 이용하면 쉽게 구현할 수 있습니다.

B. 선인장의 독립집합

- ✓ dp 식을 다음과 같이 정의해 봅시다.
- ✓ $dp(int\ n, int\ i, bool\ mc, bool\ fc)$
 - n 은 현재 cycle 번호
 - i 는 현재 cycle에서 확인하고 있는 정점 번호
 - mc 는 현재 정점이 독립집합에 속한지에 대한 변수
 - fc 는 현재 정점이 속한 cycle의 0번째 정점이 독립집합에 속한지에 대한 변수

B. 선인장의 독립집합

- ✓ $dp(\text{int } n, \text{int } i, \text{bool } mc, \text{bool } fc)$ 를 계산하는 방법.
- ✓ 우선 이웃한 자식 cycle m 에 대한 dp 값을 계산하고, 현재 계산하고자 하는 dp 값에 더해 줍니다.
- ✓ 현재 노드가 독립집합에 속한지 아닌지에 따라서 더하는 값이 다릅니다.
 - $mc = 1$ 인 경우, 이미 0 번째 노드는 독립집합에 속하기 때문에 1 번째 정점은 독립집합에 속할 수 없기에 $dp(m, 1, 0, 1)$ 을 더합니다.
 - $mc = 0$ 인 경우, 0 번째 노드는 독립집합에 속하지 않기에 1 번째 정점은 독립집합에 포함시킬 수 있기에 $\max(dp(m, 1, 0, 0), dp(m, 1, 1, 0) + 1)$ 을 더합니다.

B. 선인장의 독립집합

- ✓ $dp(\text{int } n, \text{int } i, \text{bool } mc, \text{bool } fc)$ 를 계산하는 방법.
- ✓ n 번째 cycle에 속한 $i + 1$ 번째 정점이 있는 경우, 해당 정점을 독립집합에 포함할 지를 판단하면 됩니다.
 - 다음 노드가 독립집합에 속할 수 없는 경우
 - ▶ 경우 1: i 번째 정점이 이미 독립 집합에 속한 경우
 - ▶ 경우 2: $i + 1$ 번째 정점이 0번째 정점과 이웃하면서 0번째 정점이 독립집합에 속한 경우 ($i + 1 = |C(n)| - 1, fc = 1$)
 - ▶ $dp(n, i + 1, 0, fc)$ 을 더합니다.
 - 다음 노드가 독립집합에 속할 수 있는 경우 $\max(dp(n, i + 1, 0, fc), f(n, i + 1, 1, fc) + 1)$ 을 더합니다.

B. 선인장의 독립집합

- ✓ 해당 dp 식을 통해서 최대 독립집합의 크기를 구할 수 있습니다.
- ✓ 구한 dp 값들을 통해서 역추적을 시행해서 최대 독립집합을 직접 구할 수 있습니다.
- ✓ 역추적은 이전에 나온 dp 식을 잘 활용하면 어렵지 않게 할 수 있습니다.
- ✓ 시간 복잡도 $\mathcal{O}(N)$ 에 해결할 수 있습니다.

C. 조각 체스판

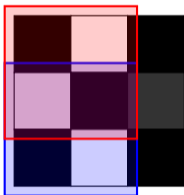
dp

출제진 의도 - **Medium**

- ✓ 제출 150번, 정답 17팀 (정답률 11.333%)
- ✓ 처음 푼 팀: **또또 나만 모르는 웰노운 낸다**@서강대 (gumgood, wbcho0504, djs100201), 29분
- ✓ 출제자: 박수현^{shiftpsh}

C. 조각 체스판

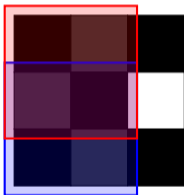
체스판은 검은색과 흰색, 또는 흰색과 검은색이 번갈아가면서 나오는 형태로 정의됩니다. 예제에서 체스판은 1×1 짜리가 9개, 2×2 짜리가 2개로 총 11개가 등장합니다.



이걸 좀 더 다루기 쉬운 형태로 바꿔서 생각할 수 있을까요?

C. 조각 체스판

체스판은 parity만이 중요하다는 관찰로부터 $(i + j) \bmod 2 = 1$ 인 모든 칸의 색깔을 뒤집어준다고 생각해 볼 수 있습니다.



이렇게 할 경우 그냥 흰색 정사각형의 수와 검은색 정사각형의 수의 합을 구하는 문제로 바뀌게 됩니다.

$f(i, j)$ 를 (i, j) 를 가장 오른쪽 아래 칸으로 갖는 흰색 정사각형의 수로 정의합시다. 그러면 f 는 다음 점화식으로 정의됩니다.

$$f(i, j) = \begin{cases} \min \{f(i-1, j), f(i, j-1), f(i-1, j-1)\} + 1 & \text{if } A_{ij} = W \\ 0 & \text{otherwise} \end{cases}$$

모든 칸에서의 f 값의 합이 흰색 정사각형 수가 됩니다.

검은색에 대해서도 똑같이 해 주고 둘을 더하면 조각 체스판의 수를 구할 수 있습니다.

D. 반짝반짝 2

math, probability

출제진 의도 – **Easy**

- ✓ 제출 99번, 정답 24팀 (정답률 24.242%)
- ✓ 처음 푼 팀: **Order of One**^{@연세대} (luciaholic, surface_03, portal3046), 7분
- ✓ 출제자: 이국렬^{1ky7674}

- ✓ 불이 들어온 기존 전구 개수의 확률변수를 X , 불이 들어온 추가 전구 개수의 확률변수를 Y 라고 합시다.
- ✓ 불이 들어온 전구 개수의 기댓값은 $E(X + Y)$ 로 표기할 수 있습니다.

Theorem: 임의의 확률변수 X, Y 에 대해서 $E(X + Y) = E(X) + E(Y)$ 를 만족한다.

- ✓ 위의 Theorem은 확률변수 X, Y 가 서로 독립이 아니어도 성립합니다.
- ✓ 때문에 $E(X)$ 와 $E(Y)$ 를 따로 계산하면 됩니다.

- ✓ i 번째 전구가 불이 들어오는지에 대한 확률변수를 X_i 로 정의합니다.

$$X_i = \begin{cases} 1, & \text{if } i\text{번째 전구가 켜진 경우} \\ 0, & \text{if } i\text{번째 전구가 꺼진 경우} \end{cases}$$

- ✓ $E(X_i) = p_i$ 를 만족합니다.

D. 반짝반짝 2

- ✓ X_i 의 정의에 따라서 확률변수 X 를 $X = \sum_{i=1}^N X_i$ 로 표현할 수 있습니다.
- ✓ 따라서 $E(X) = E\left(\sum_{i=1}^N X_i\right) = \sum_{i=1}^N E(X_i) = \sum_{i=1}^N p_i$ 를 만족합니다.

D. 반짝반짝 2

- ✓ i 번째 전구와 $i + 1$ 번째 전구 사이에 있는 추가 전구를 i 번째 추가 전구라고 합시다.
- ✓ i 번째 추가 전구에 불이 들어오는지에 대한 확률변수를 Y_i 로 정의합시다.

$$Y_i = \begin{cases} 1, & \text{if } i\text{ 번째 추가 전구가 켜진 경우} \\ 0, & \text{if } i\text{ 번째 추가 전구가 꺼진 경우} \end{cases}$$

- ✓ $E(Y_i) = p_i(1 - p_{i+1}) + p_{i+1}(1 - p_i)$ 를 만족합니다.

- ✓ Y_i 의 정의에 따라서 확률변수 Y 를 $Y = \sum_{i=1}^{N-1} Y_i$ 로 표현할 수 있습니다.
- ✓ 따라서 $E(Y) = E\left(\sum_{i=1}^{N-1} Y_i\right) = \sum_{i=1}^{N-1} E(Y_i) = \sum_{i=1}^{N-1} \{p_i(1 - p_{i+1}) + p_{i+1}(1 - p_i)\}$ 를 만족합니다.

추가 풀이

- ✓ $dp(i, j)$ 를 다음과 같이 정의하는 방법으로 $\mathcal{O}(N)$ dp로도 풀 수 있습니다.
 - i 번째 기존 전구까지만 고려했을 때, 불이 켜진 전구 개수의 기댓값.
 - j 가 1인 경우는 i 번째 기존 전구가 켜진 경우, 0은 i 번째 기존 전구가 꺼진 경우를 의미한다.
- ✓ float로는 실수 오차가 발생하기에 double을 사용해야 합니다.

E. 문자열 조작의 달인

dp

출제진 의도 - **Hard**

- ✓ 제출 57번, 정답 7팀 (정답률 12.281%)
- ✓ 처음 푼 팀: **또또 나만 모르는 웰노운 낸다**@서강대 (gumgood, wbcho0504, djs100201), 59분
- ✓ 출제자: 김규진 [snrnsidy](#)

E. 문자열 조작의 달인

- ✓ 하나의 연산을 하게 되면, 문자열의 상태가 바뀌게 됩니다. 한 번의 연산을 할 때, 바뀌게 되는 문자열의 상태들을 하나의 노드라고 생각하고, 연산들을 간선이라고 생각하면 트리구조를 떠올릴 수가 있을 것입니다.
- ✓ 따라서, 이러한 트리 구조를 전부 탐색할 수 있는 dfs와 같은 방식을 이용하여 모든 나올 수 있는 문자열의 개수를 구할 수 있습니다.
- ✓ 그러나 연산의 횟수가 최대 10^{18} 이기 때문에 이와 같은 방법으로 접근하는 것은 시간상과 메모리상의 제약으로 인해서 풀 수 없습니다.

E. 문자열 조작의 달인

- ✓ 관찰을 잘 해봅시다. 연산을 적용했을 때, 문자열이 바뀌는 경우와 바뀌지 않는 경우 2가지를 생각할 수 있습니다.
- ✓ 바뀌는 경우는 z 이외의 알파벳을 골라서 연산을 적용했을 때 바뀌게 됩니다.
- ✓ 바뀌지 않는 경우는, z 에 대하여 연산을 적용했을 때는 이전과 똑같은 것을 알 수 있습니다.
- ✓ 즉, 문자열에 대해 z 가 하나라도 있으면 z 에 대해서 남은 연산을 전부 적용한다고 했을 때, 바뀌지 않고 그대로 남아있게 됩니다.

E. 문자열 조작의 달인

- ✓ 이러한 점을 잘 캐치했다면, 이제 점화식을 세울 수 있게 됩니다. $dp[i][j][k]$ 를 다음과 같이 정의할 수 있습니다.
 - 현재 i 번째 글자까지 k 번 연산을 적용했을 때, 상태가 j 인 문자열의 개수
- ✓ 여기서 상태를 2가지 경우로 나눌 수 있는데 상태가 0인 것은 문자열에 대해서 현재 z 가 하나도 없는 경우를 의미하며, 1인 경우는 z 가 하나라도 있는 경우를 의미합니다.
- ✓ 상태를 나눈 이유는 위에서 설명했던 것처럼, z 가 하나라도 있으면 현재 요구하는 연산 이하의 횟수만큼 연산을 적용했다라도, 앞으로는 z 에 대해서만 계속 연산을 적용하면 되기 때문입니다.

E. 문자열 조작의 달인

- ✓ 문자열의 길이는 최대 300에, 문자열이 바뀐다고 했을 때 한 글자에 대해서 적용할 수 있는 연산 횟수는 a 인 경우 최대 25번 적용할 수 있습니다.
- ✓ 따라서 문자열이 변하게 되는 의미가 있는 연산 횟수는 최대 7500번입니다. (주어진 문자열이 a 로만 300개 있는 경우)
- ✓ 그리고 상태는 2가지이기 때문에 $301 \times 2 \times 7501 \times 4$ byte만큼의 사이즈만 잡고 점화식을 채워가면 됩니다.

E. 문자열 조작의 달인

- ✓ 점화식은 탐색처럼 채워나가면 되는데, 현재 글자가 $s[i]$ 라고 했을 때 변화가 있는 연산 횟수는 $z - s[i]$ 까지입니다.
- ✓ 따라서 이를 x 라고 하면 그 글자에 대해서 최대 $x - 1$ 까지 연산을 적용했을 때는 z 가 만들어지지 않고 x 번을 적용했을 때 z 가 만들어지게 됩니다.
- ✓ 이런 경우를 잘 고려하여 배열을 채워나가면 됩니다.

E. 문자열 조작의 달인

- ✓ 답을 구할 때는 2가지 경우로 나뉘는데 주어진 문자열에 대해서 전부 z 로 만들기 위해서 필요한 최소 연산 횟수를 m 이라고 하면 적용해야 하는 연산 횟수 k 보다 작거나 같은 경우, 큰 경우 2가지로 나누어 답을 구하시면 됩니다.
- ✓ k 가 m 보다 크다는 것은 결국은 전부 z 로 만들 수 있기 때문에 z 가 하나라도 안 나오는 경우는 발생하지 않습니다. 따라서 이는 $\sum_{i=0}^m dp[n][1][i]$ 가 답이 됩니다.
- ✓ 반대인 경우에는 z 가 하나도 없는 경우인 $dp[n][0][k]$ 과, z 가 하나라도 있는 경우에 대해서는 $\sum_{i=0}^k dp[n][1][i]$ 가 됩니다. 따라서 정답은 $dp[n][0][k] + \sum_{i=0}^k dp[n][1][i]$ 가 됩니다.

F. Flat Earth

math

출제진 의도 - **Easy**

- ✓ 제출 256 번, 정답 39 팀 (정답률 15.234%)
- ✓ 처음 푼 팀: **ECM**@서강대 (dart, lectura, jakads), 10분
- ✓ 출제자: 전해성^{seastar105}

- ✓ 어떤 칸에서 출발을 했을 때 지구의 끝에 가장 빨리 도달하는 방법은 제일 가까운 지구의 끝으로 일직선으로 가는 것입니다.
- ✓ 지구는 계속 커지기 때문에 자동차를 이용하지 않으면 거리를 줄일 수 없습니다.

- ✓ 자동차를 1 초 사용하면 거리를 1 만큼 줄일 수 있습니다. 따라서, 줄일 수 있는 거리는 최대 K 이며 출발 칸과 지구의 끝과의 거리가 K 보다 크다면 지구의 끝에는 도달할 수 없습니다.
- ✓ 지구의 크기가 N 일 때, 지구의 끝과 거리가 K 보다 큰 칸들은 크기가 $N - K - 1$ 이었을 때의 지구입니다.
- ✓ 따라서, 답은 크기가 N 일 때의 지구의 칸 수에서 크기가 $N - K - 1$ 일 때 지구의 칸 수를 빼준 것이 됩니다.

G. 기지국 업그레이드

greedy, segtree

출제진 의도 - **Hard**

- ✓ 제출 9번, 정답 4팀 (정답률 44.444%)
- ✓ 처음 푼 팀: **또또 나만 모르는 웰노운 낸다**@서강대 (gumgood, wbcho0504, djs100201), 97분
- ✓ 출제자: 이혜아^{ho94949}

✓ 문제 요약: 폐구간의 집합 S 에서 다음 조건을 만족하는 부분집합 T 를 찾아야 합니다.

$$\begin{aligned} & - I, J \in T \rightarrow I \cap J = \phi \\ & - \left(\bigcup_{I \in S} I \right) \subseteq \left(\bigcup_{J \in T} 3J \right) \end{aligned}$$

- ▶ 문제 처럼, 구간 J 의 중심을 X_J , 반경을 H_J 로 표현 합시다.
- ▶ $3J$ 는 중심이 X_J 이고, 반경이 $3H_J$ 인 구간입니다.

G. 기지국 업그레이드

- ✓ Greedy한 전략을 생각합시다.
- ✓ S 의 구간 I 를 반경 H_I 의 내림차순으로 순회하면서, 다음을 반복합니다.
 - H_I 가 T 에 있는 어떤 구간과도 겹치지 않으면, T 에 H_I 를 넣습니다.

G. 기지국 업그레이드

- ✓ I 를 집어넣으려고 할 때, 구간이 겹친다는 의미는 $I \cap J \neq \emptyset$ 이고 $H_I \leq H_J$ 인 J 가 이미 우리가 선택한 부분집합 T 에 존재한다는 의미입니다.
- ✓ $I \cap J \neq \emptyset$ 이면, $y \in I \cap J$ 가 존재합니다.
- ✓ $x \in I$ 에 대해,

$$|x - X_J| \leq |x - X_I| + |X_I - y| + |y - X_J| \leq H_I + H_I + H_J \leq 3H_J$$

이므로 $x \in 3J$ 입니다.

- ✓ 구간이 겹치는 경우에는, 이미 어떤 세 배 구간이 I 를 포함하고 있습니다.

G. 기지국 업그레이드

- ✓ 구간이 겹치는지 판단하는 방법은 세그먼트 트리 등을 이용하면 됩니다.
- ✓ Lazy propagation을 잘 필요 없이, 구간마다 다음 일을 반복하면 됩니다.
 1. $[X_I - H_I, X_I + H_I]$ 구간에 마킹된 노드가 있는지 체크합니다.
 2. 마킹된 노드가 없으면 $[X_I - H_I, X_I + H_I]$ 구간에 반복문을 이용해서 마킹합니다.
- ✓ 1번은 기지국의 개수만큼 진행하게 되고, 2번은 최대 세그먼트 트리의 크기만큼 진행하게 됩니다.
 - 한 번 마킹된 구간에 다시 마킹하지 않기 때문입니다.
- ✓ 좌표압축을 이용해서, 구간의 개수를 $O(|S|)$ 로 줄일 수 있습니다.
- ✓ 총 시간복잡도는 $O(|S| \log |S|)$ 이 됩니다.

H. 재활용 캠페인

sorting, greedy, two_pointer

출제진 의도 - **Medium**

- ✓ 제출 222, 정답 21팀 (정답률 9.459%)
- ✓ 처음 푼 팀: **스파르타Rebro클럽**@서강대 (pjh6792, whitehorse, 9114jin), 62분
- ✓ 출제자: 이혜아^{ho94949}

✓ 문제 요약

- 0 이상 X 이하의 수 N 개가 있고, 다음과 같은 연산을 할 수 있습니다.
- A 와 B 를 골라서 제거하고, $\min(A + B + X/2, X)$ 를 넣습니다.
- X 를 최대한 많이 만들어야 합니다.

- ✓ 관찰. 임의의 수 3개를 골라서 X 로 만들 수 있다.
 - 두 수를 합치면 수에 관계 없이 $X/2$ 이상의 수가 됩니다.
 - $X/2$ 이상인 수는 다른 어떤 수와 합쳐도 X 가 됩니다.
 - 나머지 수를 처리(?)하는 데에 이 관찰을 사용할 수 있습니다.

H. 재활용 캠페인

- ✓ X 를 최대한 많이 만드는 절차
 - 이미 존재하는 X 를 제외합니다.
 - 2개씩 합쳐서 만들 수 있는 X 를 최대한 많이 만듭니다.
 - 나머지 남은 수들을 아무렇게나 3개씩 합쳐서 X 를 만듭니다.
- ✓ 수를 2개씩 짝지어서 합이 $X/2$ 이상이 되는 쌍을 최대한 많이 만드는 문제로 바뀌었습니다.
- ✓ 이 문제는 비교적 잘 알려져 있고, $\mathcal{O}(N \log N)$ 시간에 해결할 수 있습니다.

H. 재활용 캠페인

- ✓ 관찰. 쌍을 최대한 많이 만들기 위해서 가장 큰 수를 포함하는 경우만 고려하면 됩니다.
 - 쌍이 (1 개 이상) 최대한 많이 만들어졌는데, 가장 큰 수 A 를 포함하지 않는다고 합시다.
 - 합이 X 이상이 되는 아무 쌍 (B, C) 를 고릅시다.
 - A 가 가장 큰 수 이므로, $A \geq B$ 입니다. $(A + C) \geq (B + C) \geq X/2$ 입니다.
 - (B, C) 쌍을 (A, C) 쌍으로 바꿔도, 합이 X 이상이 됩니다.
- ✓ 가장 큰 수를 고르지 않는 답이 존재할 수 있습니다.
- ✓ 하지만, 가장 큰 수를 고르면 답을 항상 찾을 수 있습니다.

H. 재활용 캠페인

- ✓ 관찰. 가장 큰 수의 짝으로는, 가능한 수중 가장 작은 수를 고르는 경우만 고려하면 됩니다.
 - 가장 큰 수 A 와 A 의 가능한 짝중 가장 작은 수인 B 가 있다고 합시다.
 - 즉, B 는 $A + B \geq X/2$ 를 만족하는 가장 작은 수입니다.
 - A 가 B 와 짝지어지지 않고, C 와 짝지어졌다고 합시다.
 - ▶ B 가 다른 수와 짝지어지지 않은 경우는 (A, C) 짝을 (A, B) 짝으로 바꿉니다.
 - ▶ 가정에 따라, $(A + B) \geq X/2$ 입니다.
 - ▶ B 가 D 와 짝지어진 경우는 $(A, C), (B, D)$ 짝을 $(A, B), (C, D)$ 짝으로 바꿉니다.
 - ▶ 가정에 따라 $C \geq B$ 이므로, $(C + D) \geq (B + D) \geq X/2$ 입니다.
- ✓ 역시, 위와 같이 짝을 고르지 않는 답이 존재할 수 있습니다.
- ✓ 하지만, 위와 같이 짝을 고르면 답을 항상 찾을 수 있습니다.

H. 재활용 캠페인

- ✓ 위 관찰을 사용해서, 다음과 같은 알고리즘을 생각할 수 있습니다.
 - 가장 큰 수 A 와 $X/2 - A$ 보다 크거나 같은 수 중 가장 작은, A 와 다른 수 B 를 찾습니다.
 - ▶ 이런 B 가 존재하지 않는다면, 더 이상 만들 수 있는 쌍이 존재하지 않습니다.
 - A 와 B 를 쌍으로 묶고, 두 수를 제외한 나머지 수에 대해 위를 반복합니다.
- ✓ 이 알고리즘을 그대로 구현하면 $\mathcal{O}(N^2)$ 시간이 나옵니다.
- ✓ multiset 등의 자료구조를 사용하면, 답을 $\mathcal{O}(N \log N)$ 시간에 찾을 수 있습니다.
- ✓ 하지만, 정렬을 제외하고 $\mathcal{O}(N)$ 시간에 답을 찾을 수 있는 방법도 있습니다. (상수도 작습니다.)
- ✓ 다음과 같은 관찰을 이용합니다.

- ✓ 관찰. 위 알고리즘으로 짝 (A, B) 와 (C, D) 가 만들어 졌을 때, $A > C$ 이면 $B \leq D$ 입니다.
 - 여기서 $A \geq B, C \geq D$ 라고 가정합니다.
 - ▶ 즉 알고리즘에서 쌍을 찾을 때 $A \rightarrow C$ 순서로 찾습니다.
 - C 가 $X/2 - C$ 보다 크거나 같은 수를 찾습니다.
 - ▶ $A > C$ 이기 때문에, $X/2 - A < X/2 - C$ 가 됩니다.
 - ▶ $X/2 - A$ 이상인 수는, $X/2 - C$ 이상인 수를 포함합니다.
 - ▶ A 가 C 보다 먼저 짝을 선택할 선택권을 가지고, 선택 가능한 수의 범위도 넓습니다.
 - ▶ 그렇기 때문에, A 의 짝인 B 가 C 의 짝인 D 보다 작거나 같습니다.

- ✓ 위 관찰을 사용해서, 알고리즘에 다음과 같은 변형을 줄 수 있습니다.
 - 남은 수 중 가장 큰 수 A 의 짝을 고를 때, 이전에 골랐던 짝보다 크거나 같은 수만 참조합니다.
- ✓ 이 경우, 같은 수를 여러번 참조하지 않기 때문에 시간복잡도는 $\mathcal{O}(N)$ 이 됩니다.

H. 재활용 캠페인

- ✓ 배열을 정렬한 뒤, 현재 남은 수가 L 번째 부터 R 번째라고 합시다.
- ✓ 최초에 $L = 1, R = N$ 으로 시작합니다.
 - L 번째와 R 번째 수의 합이 $X/2$ 이상이면, L 번째 수와 R 번째 수를 짝짓습니다.
 - ▶ 현재 남은 수 중에는 R 번째 수가 제일 큼니다.
 - ▶ 그 후, L 을 1 증가시키고, R 을 1 감소시킵니다.
 - 그렇지 않은 경우에는, 위 관찰에 따라 L 은 더 이상 사용하지 않습니다.
 - ▶ L 을 1 증가시킵니다.
- ✓ 이를 구간에서 서로 다른 두 수를 고를 수 있는 동안 (즉, $L < R$ 인 동안) 반복합니다.

I. 브런치북

implementation, parametric_search

출제진 의도 - **Hard**

- ✓ 제출 3, 정답 1팀 (정답률 33.333%)
- ✓ 처음 푼 팀: **ECM** (dart, lectura, jakads), 250분
- ✓ 출제자: 이해아^{ho94949}

I. 브런치북

✓ 문제 요약

- 다음과 같은 정렬 기준으로, 길이 N 인 모든 16진수 문자열 16^N 개를 정렬 했을 때, K 번째 문자열은?
 - ▶ 앞에서 부터 차례대로 문자를 하나씩 비교한다.
 - ▶ 비교해야 할 두 문자 중 하나 이상이 문자인 경우에는, 해당 문자로 비교한다.
 - ▶ 두 문자가 모두 숫자인 경우에는, 연속된 숫자를 묶어서 수로 생각해서 비교한다.
 - ▶ 비교해야할 수가 0을 제외하고 같은 수라면, 0의 개수가 더 많은 쪽이 크다.

I. 브런치북

- ✓ 문제를 푸는데 특별한 알고리즘이 필요하지 않습니다.
- ✓ N 이 작기 때문에, 시간 복잡도를 많이 고려할 필요도 없습니다.
- ✓ 지수시간 알고리즘은 동작하기 힘들 것 같습니다.
- ✓ 문자열의 길이가 0이거나 1인 경우, 수가 0인 경우의 예외처리를 신경써야 합니다.

I. 브런치북

- ✓ 기본적인 구현은 다음과 같습니다.
 - 모든 문자열을 적당한 분류로 묶습니다.
 - 해당 분류의 크기가 내가 찾고자 하는 등수보다 크거나 같은 경우에는, 문자열을 해당 분류에서 찾으면 됩니다.
 - 내가 찾고자 하는 등수가 해당 분류보다 큰 경우에는, 찾고자 하는 등수에서 해당 분류의 크기를 뺀 만큼을, 다음 분류부터 시작해서 찾으면 됩니다.

I. 브런치북

- ✓ 길이 N 자리의 문자열을, 처음 비교 기준인 수 혹은 문자로 줄세워보면 다음과 같습니다.
 - 숫자로 시작하고, 처음 수가 0인 문자열.
 - 숫자로 시작하고, 처음 수가 한 자리인 문자열.
 - 숫자로 시작하고, 처음 수가 두 자리인 문자열
 - ⋮
 - 숫자로 시작하고, 처음 수가 N 자리인 문자열
 - **a**로 시작하는 문자열
 - ⋮
 - **f**로 시작하는 문자열

I. 브런치북

- ✓ 숫자로 시작하고, 해당하는 수가 (0을 제외하고) K 자리인 길이 N 인 문자열을 줄세워 보면 다음과 같습니다.
 - K 자리수 X 에 대해 (0인 경우 $K = 1$ 과 같음):
 - ▶ 0이 0개 오고, X 가 오고, 영문자로 시작하는 길이 $N - K - 0$ 의 문자열.
 - ▶ 0이 1개 오고, X 가 오고, 영문자로 시작하는 길이 $N - K - 1$ 의 문자열.
 - ⋮
 - ▶ 0이 $N - K - 1$ 개 오고, X 가 오고, 영문자로 시작하는 길이 1의 문자열.
 - ▶ 0이 $N - K$ 개 오고, X 가 오는 문자열.
 - 를 K 자리 수 중 작은 수 (10^{K-1} 혹은 0) 부터 큰 수 ($10^K - 1$) 까지 반복합니다.

I. 브런치북

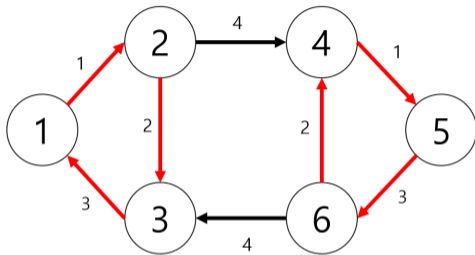
- ✓ 영문자로 시작하는 길이 i 인 문자열은 $6 \times 16^{i-1}$ 개가 있습니다.
- ✓ K 자리수 X 에 대해, 처음 수가 X 인 문자열은 $\left(1 + \sum_{i=1}^{N-K} 6 \times 16^{i-1}\right)$ 개가 있습니다.
- ✓ K 자리수 X 에 대해, X 미만의 K 자리 수는 $(X - 10^{K-1})$ 개가 있습니다.
- ✓ 정해진 영문자 (예를 들면, **a**) 로 시작하는 길이 N 인 문자열은 16^{N-1} 개가 있습니다.
- ✓ 각 분류의 크기를 알기 때문에, 분류별로 크기를 계산 해 주면 됩니다.

J. 사이클

mcmf

출제진 의도 - **Hard**

- ✓ 제출 9번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: — (—), —분
- ✓ 출제자: 이국렬^{1ky7674}

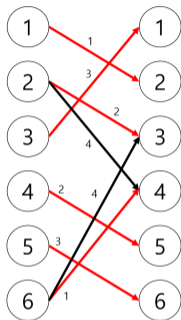


- ✓ cycle cover에 속한 간선들을 한 번 살펴봅시다.
- ✓ 각 정점에서 나가는 간선이 1개, 들어오는 간선이 1개입니다.

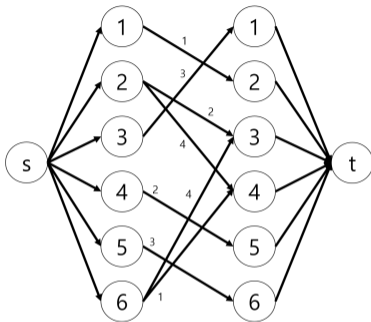
	1	2	3	4	5	6
1		1				
2			2	4		
3	3					
4					2	
5						1
6			4	3		

- ✓ cycle cover를 인접 행렬로 표현하면 열과 행이 서로 겹치지 않게 N 칸을 선택한 것과 같습니다.
- ✓ 이제 행과 열이 서로 겹치지 않게 합이 최소가 되도록 N 칸을 선택하면 됩니다.

- ✓ 행과 열이 서로 겹치지 않게 합이 최소 (or 최대) 가 되도록 N 칸을 선택하는 알고리즘.
 - min-cost max-flow 알고리즘 : 시간 복잡도 $\mathcal{O}(N^4)$
 - 헝가리안 알고리즘 : 시간 복잡도 $\mathcal{O}(N^3)$



- ✓ 이 문제는 다음과 같이 min-cost perfect matching 문제로 변환할 수 있습니다.



- ✓ min-cost perfect matching은 다음과 같은 flow 그래프에서 min-cost max-flow 알고리즘을 통해 구할 수 있습니다.

주의점

- ✓ cycle cover가 존재하지 않을 수 있습니다. (= max flow < N)
- ✓ 간선의 cost가 음수거나 0일 수 있습니다.

K. 수요응답형 버스

greedy

출제진 의도 - **Medium**

- ✓ 제출 96번, 정답 6팀 (정답률 6.25%)
- ✓ 처음 푼 팀: **또또 나만 모르는 웰노운 낸다**@서강대 (gumgood, wbcho0504, djs100201), 37분
- ✓ 출제자: 김규진^{snrnsidy}

- ✓ 배차 요청을 하는 그룹들이 N 개가 있는데, 현재 M 개의 수요응답형 버스가 운행중입니다.
- ✓ 이 때, 문제에 나온 것처럼 최대한 많이 배차 요청들을 각각의 버스에 매칭을 시켜줘야 하는 것이 이 문제의 목표입니다.
- ✓ 작년 대회에서 나왔던 마스크가 필요해와 유사하게 생각할 수 있지만 이 문제는 여러 고려해야 하는 변수들이 더 추가가 되었습니다.

K. 수용응답형 버스

- ✓ 마스크가 필요해는 가격이라는 변수에 대해서만 고려했다면 이 문제에서는 배차 시간과 인원 수 2가지 변수도 함께 고려해야 합니다.
- ✓ 2가지 변수를 한번에 고려하는 것은 복잡하기 때문에, 하나의 변수를 종속적으로 만드는 방법을 생각하는 것이 좋습니다.
- ✓ 배차 요청을 하는 그룹 관점에서 생각할 수도 있고, 수용응답형 버스 관점에서 생각할 수 있는데 저 같은 수용응답형 버스의 관점에서 생각하고 풀이를 진행하겠습니다. 또한, 인원 수를 고정시켜두고 풀이를 설명하는 쪽으로 진행하겠습니다. (사실 바뀌어도 큰 차이는 없습니다.)

K. 수요응답형 버스

- ✓ 수요응답형 버스 입장에서는 태울 수 있는 배차 요청 그룹이 어떤 경우일까요?
- ✓ 인원만 놓고 생각했을 때, 배차 요청 그룹 수 \leq 버스가 태울 수 있는 최대 인원 수가 되어야 합니다.
- ✓ 따라서, 먼저 수용응답형 버스와, 배차 요청에 대해서 인원 수를 기준으로 정렬을 시킵니다.
- ✓ 그 다음, 투 포인터 기법을 사용해야 하는데 우선 현재 수용응답형 버스에 대해서, 태울 수 있는 배차 요청을 multiset 같은 자료 구조를 이용하여 추가합니다.
- ✓ 이 때, 추가할 때는 인원 수에 대해서는 이미 확인했고 이제는 시간에 대해서만 고려하면 되므로 시간에 대한 정보를 포함하여 multiset에 추가시킵니다.
- ✓ 추가할 수 있을 때까지 추가하는데, 추가하지 못하는 경우는 그룹 인원 수가 현재 보고 있는 수용응답형 버스의 정원보다 커질 때겠죠?

K. 수요응답형 버스

- ✓ 이제 현재 보고 있는 수요응답형 버스에 대해서 multiset에 있는 배차 요청 그룹은 이미 인원 수에 대해서는 만족하기 때문에 시간만 고려해주면 됩니다.
- ✓ 즉, 이제는 변수가 시간에 대해서만 고려하면 되는 단계로 왔습니다.
- ✓ 이 때, 무작위하게 multiset중에 하나의 배차 요청 그룹을 골라서 선택하는 것이 좋을까요?
- ✓ 무작위하게 고를 때, 배차 요청 그룹이 기다리지 못하고 버스가 도착하기 전에 그냥 떠나버리는 경우가 있겠죠?
- ✓ multiset을 사용하는 이유가 여기서 나옵니다. 즉, 현재 보고 있는 버스에 대해 버스가 도착하는 시간보다 배차 요청 그룹의 기다릴 수 있는 최대 시간이 크거나 같은 경우에 대해서만 매칭시킬 수 있습니다.
- ✓ 이 때, 기다릴 수 있는 최대 시간이 버스가 도착하는 시간보다 크거나 같은 배차 요청 그룹이더라도 최대한 그 시간이 작은 그룹에 연결하는 것이 좋습니다.

- ✓ 왜 그럴까요? 잘 생각해보면 대기 가능 시간이 작은 경우 시간이 지날수록, 버스에 탈 가능성이 적어집니다. 따라서, 대기 가능 시간이 짧은 배차 요청 그룹을 먼저 처리하는 것을 생각해야 합니다. 이를 좀 더 잘 생각해보시면 무슨 말인지 이해가 갑니다.
- ✓ 따라서 이런 방식으로 매칭을 시켜나가면 최대 매칭 수를 구할 수 있게 됩니다.

L. 다꾸

segtree, case_work

출제진 의도 – **Challenging**

- ✓ 제출 0번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: — (—), —분
- ✓ 출제자: 박수현^{shiftpsh}

L. 다꾸

- ✓ 직사각형의 테두리 영역에서의 연결 요소들 중 합이 최대가 되는 연결 요소를 찾아야 합니다.
- ✓ 직사각형의 테두리 영역이 아니라 1차원 배열에 쿼리를 하는 경우 세그먼트 트리를 이용할 수 있습니다. 아래 정보 4개를 관리하는 세그먼트 트리를 만듭니다.
 - l = 맨 왼쪽 칸을 포함하는 연결 요소 중 합이 최대인 경우
 - r = 맨 오른쪽 칸을 포함하는 연결 요소 중 합이 최대인 경우
 - m = 해당 구간의 모든 부분 연결 요소들 중 합이 최대인 경우
 - s = 해당 구간의 모든 칸의 합

L. 다꾸

두 개의 노드를 아래와 같이 합칠 수 있습니다.

$$l_x \leftarrow \max \{ l_{2x}, s_{2x} + l_{2x+1} \}$$

$$r_x \leftarrow \max \{ r_{2x} + s_{2x+1}, r_{2x+1} \}$$

$$m_x \leftarrow \max \{ m_{2x}, m_{2x+1}, l_{2x} + r_{2x+1} \}$$

$$s_x \leftarrow s_{2x} + s_{2x+1}$$

KOI 2014 중등부 4번 『금광』에 등장했던 구조의 트리라서 ‘금광 트리’라고도 불립니다. 이 트리를 이용해 1차원 배열에서 문제의 쿼리를 수행할 수 있습니다.

L. 다꾸

이 퀴리를 직사각형의 테두리 영역으로 확장해 봅시다. 직사각형의 테두리를 아래와 같이 8개의 영역으로 쪼갰다고 생각할 수 있습니다.

a	b	c
d		e
f	g	h







편의를 위해 $a + b + c = B, a + d + f = D, c + e + h = E, f + g + h = G$ 라고 합시다.

L. 다꾸


- ✓ 모든 행과 모든 열에 대해 금광 트리를 만듭니다.
 - 금광 트리를 만들면 임의의 구간 안에서 (왼쪽부터 시작하는 부분 구간 중 최댓값), (오른쪽부터 시작하는 부분 구간 중 최댓값), (모든 부분 구간 중 최댓값), (구간의 합) 을 쿼리할 수 있습니다.

L. 다꾸


그러면 크게 다음과 같은 6가지 경우를 생각할 수 있습니다.

- O  모든 칸을 포함하는 경우
- G  세 모서리는 모두 포함하고, 한 모서리가 뚫려 있는 경우
- C  이웃한 두 모서리를 포함하고, 나머지 두 모서리가 뚫려 있는 경우
- U  한 모서리를 포함하고, 이웃한 두 모서리가 뚫려 있는 경우
- L  이웃한 두 모서리가 뚫려 있는 경우
- I  한 모서리가 뚫려 있는 경우


L. 다꾸

○  모든 칸을 포함하는 경우

답은 자명하게 $a + b_s + c + d_s + e_s + f + g_s + h$ 입니다.

I  한 모서리가 뚫려 있는 경우

위, 아래, 왼쪽, 오른쪽을 고려하면 $\max \{B_m, D_m, E_m, G_m\}$ 입니다.


L  이웃한 두 모서리가 뚫려 있는 경우

왼쪽 위, 왼쪽 아래, 오른쪽 위, 오른쪽 아래를 고려하면

$$\max \{B_l + D_l - a, B_r + E_l - c, \\ D_r + G_l - f, E_r + G_r - h\}$$


입니다.

L. 다꾸

- U  한 모서리를 포함하고, 이웃한 두 모서리가 뚫려 있는 경우
위, 아래, 왼쪽, 오른쪽을 고려하면

$$\max \{b_s + D_l + E_l, B_l + d_s + G_l, \\ B_r + e_s + G_r, D_r + E_r + g_s\}$$


입니다.

- C  이웃한 두 모서리를 포함하고, 나머지 두 모서리가 뚫려 있는 경우
왼쪽 위, 왼쪽 아래, 오른쪽 위, 오른쪽 아래를 고려하면


$$\max \{B_r + d_r + e_s + f + g_s + h, b_r + c + D_r + e_s + g_s + h, \\ B_l + d_s + e_r + f + g_s + h, a + b_l + d_s + E_r + f + g_s, \\ b_s + c + D_l + e_s + g_r + h, a + b_s + c + d_l + e_s + G_r, \\ a + b_s + d_s + E_l + f + g_l, a + b_s + c + d_s + e_l + G_l\}$$

입니다.

L. 다꾸

- G  세 모서리는 모두 포함하고, 한 모서리가 뚫려 있는 경우 $X_l + X_r$ 을 하면 중간이 겹칠 수 있기 때문에, X_s 에서 (부분 구간 중 합의 **최솟값**) 을 빼 줘야 합니다.
- ✓ 모든 원소에 -1 을 곱하고 합의 최댓값을 구한 뒤, 여기에 다시 -1 을 곱하면 원래 원소들의 합의 최솟값이 됩니다.
 - ✓ 다른 방법으로도 합의 최솟값을 관리할 수 있습니다.

이 값을 X_n 이라고 합시다.

- G  세 모서리는 모두 포함하고, 한 모서리가 뚫려 있는 경우 모든 칸의 합을 S 라고 합시다. 그러면 답은 $\max \{S - B_n, S - D_n, S - E_n, S - G_n\}$ 입니다.

모든 6개의 경우 중 최댓값이 쿼리의 답이 됩니다.