SUAPC 2020 풀이

Official Solutions

by

SUAPC 2020 출제진

2020년 8월 15일

문제			의도한 난이도	출제자	
2A		가뭄 (Small)	**Easy**	이국렬 연세대	lky7674
2B		Polynomial	**Easy**	이재열 홍익대	malkoring
2C	1E	독특한 계산기	**Medium**	이국렬 연세대	lky7674
2D		비드맨	**Hard**	김규진 서강대	snrnsidy
2E		두 번째 트리의 지름	**Hard**	이국렬 연세대	lky7674
2F		싸이버개강총회	**Easy**	이재열 홍익대	malkoring
2G		약수	**Hard**	정연두 홍익대	Green55
2H		난개발	**Hard**	박수현 서강대	shiftpsh
2I	1G	마스크가 필요해	**Challenging**	김규진 서강대	snrnsidy
2J	1J	상품권 준비	**Challenging**	정연두 홍익대	Green55
2K		객실 배치	**Hard**	이재열 홍익대	malkoring
2L	1A	200년간 폐관수련했더니...	**Medium**	정연두 홍익대	Green55

문제			의도한 난이도	출제자	
1A	2L	200년간 폐관수련했더니...	Easy	정연두 홍익대	Green55
1B		꿀벌	Challenging	박수현 서강대	shiftpsh
1C		케이크 커팅	Hard	이준석 서강대	semteo04
1D		가뭄 (Large)	Challenging	이국렬 연세대	lky7674
1E	2C	독특한 계산기	Easy	이국렬 연세대	lky7674
1F		울타리	Hard	이준석 서강대	semteo04
1G	2I	마스크가 필요해	Medium	김규진 서강대	snrnsidy
1H		전설	Medium	박수현 서강대	shiftpsh
1I		수학은 재밌어	Easy	김규진 서강대	snrnsidy
1J	2J	상품권 준비	Medium	정연두 홍익대	Green55
1K		물건 가져가기	Hard	이국렬 연세대	lky7674
1L		카드 셔플	Challenging	이준석 서강대	semteo04

2A. 가뭄 (Small)

`math`

출제진 의도 – **Easy**

- 제출 41번, 정답 26팀 (정답률 63.42%)
- 처음 푼 팀: **Terra**[서강대학교] (이민희, 이동주, 조원빈), 5분
- 출제자: `lky7674`

2A. 가뭄 (Small)

- 0솔 방지를 위한 단순 연립방정식 풀기 입니다.

- $a = \frac{d_1 + d_2 - d_3}{2}, b = \frac{d_1 + d_3 - d_2}{2}, c = \frac{d_2 + d_3 - d_1}{2}$.

- d_i 가 자연수이기 때문에 a, b, c는 무조건 정수$/2$ 입니다.
- 연립방정식을 풀었을 때, a, b, c 전부 0보다 큰 것을 체크해야 합니다.

2B. Polynomial

`math`

출제진 의도 – **Easy**

- 제출 108번, 정답 19팀 (정답률 17.59%)
- 처음 푼 팀 : **3vs500**[서강대학교] (장수길, 권형준, 강민석), 11분
- 출제자 : `malkoring`

2B. Polynomial

- 원래 의도는 exponentiation + Horner's rule 을 의도하고 small/large 나누려고 했으나.....
- 지금은 이렇게 단순히 반복문만 돌리면 되는 문제가 되었습니다!

- 문제에서 소개한 것처럼, Horner's rule은 다항식을 빠르게 evaluate하는 알고리즘입니다.
- 하지만, 위의 배경지식이 없더라도 문제는 쉽게 풀 수 있게 입력의 조건을 단순하게 잡았습니다.

2B. Polynomial

- 다항식을 구성하는 각 항은 $N+1$ 개씩 입력으로 주어집니다.
- 계수가 0 이더라도 명시적으로 무조건 입력으로 들어오게 됩니다.
- 모든 항의 차수는 $N, N-1, N-2, N-3, \ldots, 1, 0$
  이렇게 등차수열인 것이 보장되기 때문에, 내림차순으로 정렬을 할 필요도 없습니다.

- 나머지 연산은 덧셈/곱셈에 대해서 결합법칙이 성립하기 때문에, 덧셈/곱셈을 수행해 줄 때마다 나머지 연산을 해주면 됩니다.

– 따라서, 문제에서 설명한 것처럼 x를 곱하고, 계수를 더하고, 나머지를 취해주는 작업을 반복문을 돌려서 해주시면 됩니다.

2F. 싸이버개강총회

`tree_set`, `hash_set`
출제진 의도 – **Easy**

- 제출 90번, 정답 9팀 (정답률 10.00%)
- 처음 푼 팀: **한놈만노린다**서강대학교 (최재영, 이수빈, 황성민), 63분
- 출제자: `malkoring`

2F. 싸이버개강총회

- i 번째 줄에서 채팅로그가 찍힌 시간을 T_i, 채팅로그에 적힌 닉네임을 N_i 라고 합시다.
- $T_i \leq S$ 일 때 나타나는 N_i,
  그리고 $E \leq T_j \leq Q$ 일 때도 나타나는 N_j $(= N_i)$ 를 찾으면 됩니다. $(i \neq j)$

- 찾는 방법은 간단합니다. 여기서 채팅로그는 10만줄까지 입력될 수 있기 때문에, 브루트포스로는 풀기 어려울 수 있습니다.

- C++의 경우, `std::map` / `std::set` 컨테이너를 이용해서
- Java/Kotlin의 경우, `HashSet` / `HashMap` / `TreeSet` / `TreeMap`을 이용해서
- Python의 경우, dictionary를 이용해서 비교적 간단하게 카운팅이 가능합니다.

위에서 설명한 방법대로, S 이전에 입장한 것으로 파악되는 인원 $\&$ E 이후 Q 이전에 퇴장한 것으로 파악되는 인원을 카운팅하면 됩니다.

1A/2L. 200년간 폐관수련 했더니 PS 최강자가 된 건에 대하여

`greedy`

출제진 의도 – **Easy**Div 1 / **Medium**Div 2

D1 제출 149번, 정답 16팀 (정답률 10.74%)

처음 푼 팀: **개강시켜조**연세대학교 (김형진, 함태완, 동용훈), 10분

D2 제출 55번, 정답 5팀 (정답률 9.09%)

처음 푼 팀: **입영통지서 필 무렵**서강대학교 (한성환, 안성훈, 강효규), 77분

– 출제자: `Green55`

1A/2L. 200년간 폐관수련 했더니 PS 최강자가 된 건에 대하여

- 일단 순서대로 모든 대회를 쭉 참가해 봅시다.
- 성공했다면, 행복하게 상금을 독식하면 됩니다.

- 실패했다면, 처음으로 상금 제한에 걸려 참가가 불가능해진 대회가 i 번째 대회라고 해봅시다.
- 가장 먼저 시도해 볼 수 있는 것은, i 번째 대회를 제외하고 모든 대회를 참가해 보는 것입니다.
- 성공했다면, 행복하게 (i 번째 대회만 빼고) 상금을 독식하면 됩니다.

1A/2L. 200년간 폐관수련 했더니 PS 최강자가 된 건에 대하여

실패했다면..

- i번째 대회를 참가하지 않는 것은 실패했으니, i번째 대회를 **참가 할 수 있도록** 만들어 야합니다.
- i번째 대회 이전에 열린 대회를 하나 골라, 그 대회에 불참해 상금의 합을 줄여봅시다.
- 상금을 최대한 많이 줄일수록 좋으니까, i번째 대회 이전에 열리는 대회 중
- **받는 상금이 제일 큰 대회만** 골라서 불참해봐도 충분합니다.
- 이렇게까지 했는데도 실패했다면 연두는 냉동되어야 합니다.

따라서 $\mathcal{O}(N)$이 걸리는 시뮬레이션을 3번만 해서 답을 구할 수 있습니다.

1E/2C. 독특한 계산기

`parsing, deque`

출제진 의도 – **Easy**[Div 1] / **Medium**[Div 2]

D1 제출 29번, 정답 9팀 (정답률 31.03%)

처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 29분

D2 제출 36번, 정답 3팀 (정답률 8.33%)

처음 푼 팀: **기시디한테 롤 진 팀**[홍익대학교] (노정윤, 김도현, 김연욱), 54분

– 출제자: `1ky7674`

1E/2C. 독특한 계산기

Deque는 맨 앞의 원소, 또는 맨 뒤의 원소를 $\mathcal{O}(1)$에 제거할 수 있는 자료 구조입니다.

- 2개의 Deque를 만들고, 문자열을 읽어내면서 하나에는 수를 다른 하나에는 부호를 넣습니다.
  - 문자열을 Parsing 할 때, 숫자를 읽으면 (지금까지 읽은 수) $\times$ 10 + (현재 읽고 있는 숫자)로 지금까지 읽은 숫자를 갱신하고, 연산자를 읽으면 지금까지 읽은 수와 연산자를 Deque에 넣으면 된다.
  - 앞에 0이 붙어 있는 것, 숫자가 0인 것, 맨 앞의 문자가 마이너스 기호인 경우를 주의하면서 구현해야 합니다.
- 앞의 원소 2개로 계산한 것, 뒤의 원소 2개로 계산한 것들끼리 비교하면서 식을 계산해 나갑니다.

11. 수학은 재밌어

`euler_phi`, `number_theory`, `math`
출제진 의도 – **Easy**

- 제출 64번, 정답 10팀 (정답률 15.63%)
- 처음 푼 팀 : **처음보는사람들끼리신청마감1시간전에만든팀**[서강대학교] (김성현, 전해성, 박재형), 10분
- 출제자 : `snrnsidy`

- 이 문제를 풀기 위해서는 오일러 피 함수의 계산 방법을 알아야 합니다.
- 오일러 피 함수의 특징은 다음과 같습니다.
  1. 만약 두 정수 m, n 이 서로소라면 다음을 만족한다. $\phi(mn) = \phi(m)\phi(n)$
  2. 정수 p 가 소수인 경우에 $\phi(p) = p - 1$ 이다.
  3. 소수 p 의 거듭제곱 p^x 의 오일러 피 함수 값 $\phi(p^x)$ 는 $p^{x-1}(p-1)$ 입니다.
- 이를 통해서 오일러 피 함수값을 구하려면 소인수분해를 해야 함을 알 수 있습니다.

- 소인수분해를 하기 위해서는 소수들을 미리 구할 필요가 있습니다. 에라토스테네스의 체를 이용해서 구하면 됩니다.
- 하지만, 10^9 까지의 소수를 모두 구할 경우 시간이 너무 오래 걸립니다. 에라토스테네스의 체의 시간 복잡도는 $\mathcal{O}(n \log \log n)$ 입니다.

그래서 봐야 하는 범위를 줄일 필요가 있습니다. 여기서는 $\sqrt{10^9}$ 까지만 보면 됩니다.
왜냐하면...

- 합성수 x 는 ab와 같은 두 정수의 곱으로 표현할 수 있습니다. 이 때 $a \leq b$라고 한다면 $x = ab \leq b^2$ 이므로, $b \geq \sqrt{x}$ 가 됩니다. 따라서, b의 최소값은 $\sqrt{x}$ 이고, a의 최대값은 $\sqrt{x}$ 가 됩니다.
- 그러므로 x 가 합성수라면 $\sqrt{x}$ 보다 작거나 같은 a와 $\sqrt{x}$ 보다 크거나 같은 b의 곱으로 이루어지게 됩니다. 따라서 $\sqrt{x}$ 까지만 봐도 충분합니다.

이제 a를 1부터 하나씩 증가시키면서 x가 a로 나누어 떨어지는 경우를 찾습니다.

- 이 때 $b = x/a$라고 두면, $\phi(a)$의 값을 구해서 b와 같은지를 비교해주면 됩니다. 또한 $\phi(b)$의 값을 구해서 a와 같은지도 비교해 줍니다.
- 하지만 반복문을 돌릴 때 10^9까지 돌리면 당연히 시간 초과가 나오게 됩니다. 위에서 언급한 대로, a는 $\sqrt{x}$까지만 확인해도 충분합니다.

- 이렇게 해서 최소값이 존재할 경우 그 값을 출력하면 되고, 아니라면 -1 을 출력하면 됩니다.
  - 최소값을 구할 때 초기값을 충분히 큰 값으로 잡는다면, 반복문이 끝나고 나서도 최소값이 초기값과 같은 경우 $x\phi(x) = n$ 인 x 가 존재하지 않음을 알 수 있습니다.

2G. 약수

dp

출제진 의도 – **Hard**

- 제출 17번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: –
- 출제자: Green55

2G. 약수

2020/08/04 와우약수터 사전 답사 후기 : 더웠습니다

2G. 약수

- 1 은 모든 양의 정수의 약수입니다.
- 와우매직을 쓸거면 무조건 1 로 바꿔줍니다.
- 와우매직을 쓰지 않은 미네랄의 함량이 약수 - 배수라면, 약수입니다!
- 1 이 싫으면 $2 \times 3 \times \cdots \times 10^9$ 로 바꿔도 됩니다. (모든 양의 정수의 배수)

- 이제 N 개의 원소 중 약수 - 배수 관계인 원소 k 개를 최대한 많이 골라주고
- 나머지 $N - k$ 개의 원소에 와우매직을 사용하여 1 로 바꿔주면 됩니다.

2G. 약수

- 모든 미네랄 성분들의 함량을 오름차순으로 정렬한 배열을 $a[]$ 라고 하면
- $dp[i]$ = 다음 조건을 만족하는 집합 S_i 의 최대 크기
  1. $a[1], a[2], \ldots, a[i]$ 의 부분집합
  2. $a[i]$ 를 반드시 포함
  3. 모든 원소들이 약수-배수 관계

2G. 약수

- 집합 S_j 에서 제일 큰 원소는 $a[j]$ 입니다.
- 만약 $a[i]$ 가 $a[j]$ 의 배수라면, $a[i]$ 는 S_j 의 모든 원소의 배수입니다.
- 따라서 S_j 에 $a[i]$ 를 추가하여 더 큰 집합을 만들 수 있고, 다음과 같은 점화식이 유도됩니다.
- $dp[i] = \max(1, \max_j(dp[j] + 1))$ (단, $j < i$ 이고 $a[j]$ 는 $a[i]$ 의 약수)
- 1 이 $a[i]$ 로만 집합을 만드는 경우이고, $(dp[j] + 1)$ 이 S_j 에 $a[i]$ 를 합치는 경우입니다.
- 답은 $N - \max_i dp[i]$ 이며, 총 $\mathcal{O}\left(N^2\right)$ 에 계산 할 수 있습니다.

2E. 두 번째 트리의 지름

`tree`, `dfs`

출제진 의도 – **Hard**

- 제출 26번, 정답 1팀 (정답률 3.85%)
- 처음 푼 팀: **삼지선다**연세대학교 (박선종, 이다현, 이지수), 98분
- 출제자: `lky7674`

2E. 두 번째 트리의 지름

- 연세대학교 안형찬 교수님의 알고리즘 분석에서 나왔던 문제의 아이디어를 하나 가져왔습니다.
- 대회 중에 tree dp로 푸신 분이 있었는데, 이 풀이는 div2에 적절한 풀이라고 생각되지 않아서 포함하지 않았습니다.

2E. 두 번째 트리의 지름

BOJ 1967 트리의 지름 문제에서 트리의 지름을 구하는 방법을 알아봅시다.

- 처음에는 주어진 트리에서 트리의 지름을 구하면서 해당 양 끝 정점 A, B 를 구합니다.
- A 를 제외한 상태에서 트리의 지름 R_1 을 구합니다.
- B 를 제외한 상태에서 트리의 지름 R_2 를 구합니다.
- $\max(R_1, R_2)$ 가 두 번째 트리의 지름이 됩니다.

2E. 두 번째 트리의 지름

간단한 증명:

- R_1은 $d(A, B)$와 $\{x \neq B : d(A, x)\}$를 제외한 거리들 중 최댓값입니다.
- R_2은 $d(A, B)$와 $\{x \neq A : d(x, B)\}$를 제외한 거리들 중 최댓값입니다.
- 따라서 $\max(R_1, R_2)$는 $d(A, B)$를 제외한 거리들 중 최댓값입니다.

2H. 난개발

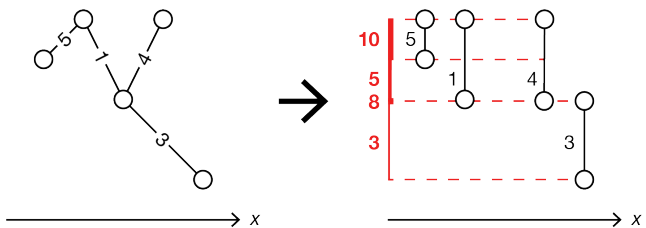
`sweeping`

출제진 의도 – **Hard**

- 제출 13번, 정답 1팀 (정답률 7.69%)
- 처음 푼 팀: **Terra**[서강대학교] (이민희, 이동주, 조원빈), 105분
- 출제자: `shiftpsh`

2H. 난개발

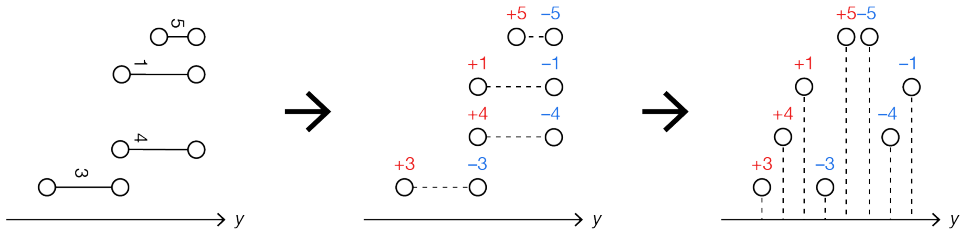
도로들을 y 축에 사영시키고 겹치는 도로들의 통행량을 전부 합친다고 생각해 봅시다. 예제의 경우 이렇게 됩니다.



그러면 이 문제는 가중치가 있는 선분들을 겹쳐서 그 중 값이 최대가 되는 구간을 찾는 문제가 됩니다.

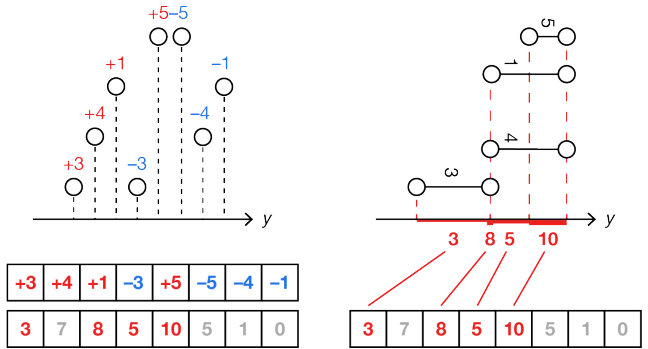
2H. 난개발

이 문제는 스위핑으로 해결 가능합니다. 선분의 시작점과 끝점을 아래 그림과 같이 값이 존재하는 각각의 점으로 생각하고, y 좌표가 작은 순으로, y 좌표가 같다면 값이 큰 순으로 정렬합니다. 값이 큰 순으로 정렬하는 이유는 후술합니다.



2H. 난개발

이렇게 정렬된 점들을 배열의 원소라고 생각하고 왼쪽부터 차례로 보면서 prefix sum을 구하면, 모든 구간에 대해 '겹치는 도로들의 통행량을 전부 합친' 값을 구할 수 있습니다.



왼쪽에서 오른쪽으로 마치 직선이 쓸고 지나가는 것처럼 누적하면서 각 구간의 합을 구하는 기법이라 '스위핑' 이라고 불립니다.

값이 큰 순으로 정렬하는 이유는 도로의 양 끝 점도 도로로 간주하기 때문인데, 스위핑을 하면서 '더해지는' 정점이 '빠지는' 정점보다 먼저 처리되기를 원하기 때문입니다. '더해지는' 정점이 먼저 처리되어야 같은 y 좌표에서 여러 시종점들이 겹치는 경우를 제대로 처리할 수 있습니다.

위에서 얻은 prefix sum 배열에서의 최댓값이 문제의 정답이 됩니다.

2K. 객실 배치

`dp`, `exponentiation_by_squaring`
출제진 의도 – **Hard**

- 제출 22번, 정답 1팀 (정답률 4.56%)
- 처음 푼 팀: **Terra**서강대학교 (이민희, 이동주, 조원빈), 92분
- 출제자: `malkoring`

2K. 객실 배치

- 성민이는 객실을 상하좌우로 인접하지 않게 고객을 호실로 안내해야 합니다.
- 같은 층에 고객을 배치하는 경우는 고려할 수 없습니다.
- 위층/아래층으로 인접하지 않게 고객을 배치해야 하기 때문에
  아래층인 101호, 바로 위층인 201호에 고객을 같이 배치할 수 없습니다.

2K. 객실 배치

- 먼저 층수가 1층인 경우에 대해서 케이스를 나눠봅시다.
  1. 고객을 배치하지 않을 수도 있습니다.
  2. 101호에만 고객을 배치할 수 있습니다.
  3. 102호에만 고객을 배치할 수 있습니다.

- 1층 호텔인 경우에는, 이렇게 3가지 경우의 수가 나옵니다.

2K. 객실 배치

- 그럼, 이제 층수가 2 층인 경우에 대해서 케이스를 나눠봅시다.
  1. 101 호에 고객이 배치되어 있다면, (2 가지)
     - 인접하지 않은 202 호에 고객을 배치하거나 혹은 아예 배치하지 않아도 됩니다.
  2. 102 호에 고객이 배치되어 있다면, (2 가지)
     - 인접하지 않은 201 호에 고객을 배치하거나 고객을 아예 배치하지 않아도 됩니다.
  3. 1 층에 고객을 배치하지 않았다면, (3 가지)
     - 201 호, 202 호 어디에든 고객을 배치해도 되고 (2 가지) 똑같이 아예 배치하지 않아도 됩니다.(1 가지)
- 따라서, 2 층 호텔인 경우에는, 이렇게 7 가지 경우의 수가 나옵니다.

- 그럼 위에서 도출한 결과를 가지고 일반화된 식을 유도해 보겠습니다.
  1. (N)01 호에 고객을 배치할 수 있으려면,
     - ($N-1$)02 호에 고객이 배치되어 있거나, ($N-1$) 층에 고객이 배치되지 않아야 한다.
  2. (N)02 호에 고객을 배치할 수 있으려면,
     - ($N-1$)01 호에 고객이 배치되어 있거나, ($N-1$) 층에 고객이 배치되지 않아야 한다.
  3. (N) 층에 고객을 배치하지 않으려면, 앞의 어떤 3 가지 조건이든 상관없다.

- N 층에 객실을 배치하려면 $N-1$ 층에 객실을 어떻게 배치했는지만 알면 됩니다.
- $N+1$ 층도 마찬가지겠죠.

2K. 객실 배치

- 앞에서의 설명을 정리해서 식으로 정리하면 아래와 같이 나타낼 수 있습니다.

$$Room_{N,01} = Room_{N-1,02} + Room_{N-1,00} \quad (1)$$

$$Room_{N,02} = Room_{N-1,01} + Room_{N-1,00} \quad (2)$$

$$Room_{N,00} = Room_{N-1,01} + Room_{N-1,02} + Room_{N-1,00} \quad (3)$$

- 어떻게 DP로 비벼볼 수는 있을 것 같습니다.
- 하지만 그냥 DP로는 어림도 없습니다. 왜냐면, N이 최대 10^{18}이기 때문이에요.

2K. 객실 배치

- 이럴 때 쓰는 것이 뭐다?
- 그것은 바로 행렬 거듭 제곱
- 위에서 유도한 DP 점화식을 다른 방식으로 나타내봅시다

- 예를 들면, DP 점화식을 행렬로 나타낸다면 어떨까?

위의 점화식을 행렬로 나타내면 아래와 같습니다.

$$\begin{pmatrix} Room_{2,01} \\ Room_{2,02} \\ Room_{2,00} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} Room_{1,01} \\ Room_{1,02} \\ Room_{1,00} \end{pmatrix}$$

– 그렇다면, N 번째 항을 구하려면 어떻게 해야할까요?

2K. 객실 배치

그냥 행렬을 N 번 거듭제곱해 주시면 됩니다!

$$\begin{pmatrix} Room_{N,01} \\ Room_{N,02} \\ Room_{N,00} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^{N} \begin{pmatrix} Room_{0,01} \\ Room_{0,02} \\ Room_{0,00} \end{pmatrix}$$

- 하지만... N 은 10^{18} 이고 $M^3 * N$ 은 너무 느리지 않을까요..?
- 행렬의 거듭제곱은 빨리 구할 수 있는 방법이 이미 있죠!

2K. 객실 배치

- 행렬의 거듭제곱끼리 곱하면 지수의 합의 법칙이 성립하기 때문에,
- 행렬의 2^0 제곱, 2^1 제곱, 2^2 제곱, ..., 2^{60} 제곱을 모두 구해서 행렬의 N 제곱을 구하는 데에 필요한 만큼 곱해주면 됩니다.

- 예를 들면, 19930919 만큼의 거듭제곱을 구한다고 가정해 봅시다.
- 19930919 는 이진법으로 $1001100000001111100100111_2$ 로 나타낼 수 있습니다.
- 이를 2 의 거듭제곱의 합으로 나타내면, $2^0 + 2^1 + 2^2 + 2^5 + 2^8 + 2^9 + 2^{10} + 2^{11} + 2^{12} + 2^{20} + 2^{21} + 2^{24}$ 가 되는 것이죠.

2K. 객실 배치

- 행렬의 거듭제곱에도 똑같이 적용할 수 있습니다.
- 그럼 19930919 번째 층에 객실을 배치하려면?

$$\begin{pmatrix} Room_{19930919,01} \\ Room_{19930919,02} \\ Room_{19930919,00} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^{2^0+2^1+2^2+2^5+2^8+\cdots+2^{21}+2^{24}} \begin{pmatrix} Room_{0,01} \\ Room_{0,02} \\ Room_{0,00} \end{pmatrix}$$

2K. 객실 배치

- 위의 방식처럼 $Room_{N,01}$, $Room_{N,02}$, $Room_{N,00}$ 를 구한 다음, 합하시면 정답을 구할 수 있습니다.
- 따라서, $\mathcal{O}\left(M^3 \log N\right)$ 안에 간단히 해결할 수 있습니다.

2D. 비드맨

`greedy`
출제진 의도 – **Hard**

- 제출 74번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: —
- 출제자: `snrnsidy`

2D. 비드맨

문제를 봤을 때 만화 비드맨이 생각나실 수도 있겠지만 사실 만화랑은 관련이 없는 문제입니다.
이 문제는 그리디하게 접근하면 풀 수 있습니다.

- 값 n 개의 배열 $x_1, x_2, x_3, ..., x_n$ 가 있다고 할 때, 어떤 2 개의 값을 선택합니다.
- 그 후 큰 값에서 작은 값을 빼줘서 그 값이 0 보다 클 때는 다시 배열에 추가합니다. (배열의 크기가 1 줄어듭니다.)
- 이 동작을 반복했을 때 결국은 하나의 값만 남게 되는데, 이 값을 최소로 만들어주면 됩니다.

값의 범위가 크기 때문에 브루트 포스로는 시간 초과를 받습니다.

2D. 비드맨

- 먼저 입력 받은 값들을 오름차순으로 정렬합니다. 그 후 배열의 값들의 합을 구합니다.
- 그리고 배열의 최댓값을 구합니다. 배열을 정렬했으므로 최댓값은 맨 뒤에 있습니다.
- 배열의 합과 최댓값을 구했다면 이 둘의 관계에 따라서 접근 방법이 달라집니다. 배열의 합은 S로 배열의 최댓값은 M으로 표기하겠습니다.
  1. $S - M < M$일 때
  2. $S - M \geq M$일 때

1. $S - M < M$ 일 때:

- 최댓값이 나머지 값들의 합보다 큰 경우로써 이 때는 나머지 값들 전부를 최댓값에서 빼줘야 합니다.
- 나머지 값들 중에서 2 개를 고르게 된다면 후에 최댓값을 좀 더 작게 만들 수 없게 됩니다.

1. $S - M < M$ 일 때:

- 예를 들자면, 배열이 $\{2, 3, 7\}$ 이라고 합시다. 이 때 $2 + 3 = 5 < 7$ 입니다.
- 만약 여기서 2와 3을 고르게 되면 1이 들어가게 되고, 이 1과 7을 고르면 6이 남게 됩니다.
- 이는 최소 값인 2보다 큰 값이므로 나머지 값들에서 2개를 고르게 된다면 절대 최적의 값을 구할 수 없습니다.
- 그러므로 최댓값을 무조건 고르고 다른 하나는 나머지 값들을 고르는 것이 답이 됩니다. 즉 $2M - S$가 답이 됩니다.

2. $S - M \geq M$ 일 때

- 다행히도 값들의 합이 홀수인지 짝수인지에 따라서 최소 값을 구할 수 있습니다.
- 값들의 합이 홀수라면 위처럼 최적으로 진행을 했을 때 무조건 하나는 남게 됩니다. 그리고 짝수라면 다 매칭을 할 수 있게 되어서 아무것도 안남게 됩니다.
- 그러므로 S 이 2 로 나눠지면 (짝수이면) 0 을, 안나눠지면 (홀수이면) 1 을 출력하면 됩니다.

1G/2I. 마스크가 필요해

`greedy`

출제진 의도 – **Medium**[Div 1] / **Challenging**[Div 2]

D1 제출 19번, 정답 2팀 (정답률 10.53%)

처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 68분

D2 제출 22번, 정답 0팀 (정답률 0.00%)

처음 푼 팀: —

– 출제자: `snrnsidy`

1G/2I. 마스크가 필요해

이 문제는 입력 값들의 범위가 작은 경우에는 이분 매칭으로 풀 수 있는 문제입니다.

- 하지만 A도시의 사람과 상점의 개수가 최대 $500,000$이기 때문에 이분 매칭으로 풀려고 하면 시간 초과가 나오게 됩니다.
- 결론부터 말하자면 그리디로 접근해야 합니다.

1G/2I. 마스크가 필요해

예를 들어...

- 현재 A 도시에는 $[L_1, R_1] = [1, 4]$ 인 사람과 $[L_2, R_2] = [3, 6]$ 인 사람이 있습니다. 그리고 A 도시의 상점은 2 군데가 있는데 $(P_1, X_1) = (3, 1)$ 인 상점과 $(P_2, X_2) = (5, 1)$ 인 상점이 있습니다.
  - 만약 여기서 상점 1 에서 2 번째 사람이 마스크를 구매하면, 1 번째 사람은 마스크를 구할 수 없습니다.
  - 그러나 상점 1 에서는 첫 번째 사람이, 상점 2 에서는 두 번째 사람이 구매를 하면 2 명 다 마스크를 구할 수 있게 됩니다.
- 즉, 마스크의 가격이 작은 상점부터 보면서 마스크를 구매할 수 있는 사람들에 대해서 R 값이 작은 사람들부터 먼저 하나씩 매칭을 시켜주면 최적의 해를 구할 수 있게 됩니다.

1G/2I. 마스크가 필요해

- 먼저 어떤 상점 Y 가 파는 마스크를 A 도시의 사람들이 살 수 없다면 (즉, $L \leq P_Y \leq R$ 를 만족하는 사람이 존재하지 않다면) 그 때는 그 상점에서 아무도 구매할 수 없기 때문에 무시합니다.
- 위의 경우가 아닌 경우에 대해서, 방금 언급한 것처럼 상점 Y 에서 마스크를 구매할 수 있는 사람들 중에서 R 의 값이 가장 작은 사람을 X 라고 합시다.
- 사람 X 가 상점 Y 에서 마스크를 구매하는 것이 최적의 해를 얻을 수 있게 되는데 사람 X 가 상점 Y 에서 마스크를 구매하지 않는 것이 최적의 해 S 를 구할 수 있다고 가정합시다. 이 경우에 대해서 최적의 해를 얻을 수 없음을 보임으로써 사람 X 가 상점 Y 에서 마스크를 구매하는 것이 최적임을 보이려고 합니다.

1G/2I. 마스크가 필요해

1. **사람 A와 상점 B가 매칭이 되어 있지 않은 경우**: A와 B를 매칭할 수 있기 때문에 한 사람이 더 마스크를 얻을 수 있습니다. 따라서 최적해라고 볼 수 없습니다.
2. **사람 A만 매칭이 되어 있는 경우**: 사람 A를 상점 B에 매칭시켜주면서 기존에 A에 매칭되어 있던 상점 B'을 다른 사람에게 매칭시켜주면 더 많은 사람이 마스크를 가질 수 있게 됩니다.
3. **상점 B에만 매칭이 되어 있는 경우**: 2번째 경우와 같습니다.
4. **사람 A는 상점 B'에, 상점 B는 사람 A'에 매칭이 되어 있는 경우**: 사람 A를 상점 B에 매칭시키면 사람 A'은 상점 B'에 매칭이 되거나 다른 상점 B''에 매칭됩니다. 즉, 이 경우에도 더 많은 사람이 마스크를 가질 수 있게 됩니다.

   그러므로 사람 A에 상점 B가 매칭이 되어 있는 경우가 최적의 해임을 알 수 있습니다.

1G/2I. 마스크가 필요해

- 마스크는 가격에 대해서 오름차순으로 정렬합니다.
- 사람은 L에 대해서 오름차순으로 정렬하되 L이 같은 경우에는 R값이 작은 사람이 먼저 오도록 합니다. 우선순위 큐로 관리합니다.

1G/2I. 마스크가 필요해

- 마스크를 작은 가격부터 큰 가격 순서대로 하나씩 보면서, 마스크의 가격 범위에 해당되는 사람을 추가할 수 있으면 추가해줍니다. 또한 현재 마스크의 가격이 우선순위 큐의 맨 위의 값보다 클 경우, 즉 마스크를 살 수 없는 경우에는 우선순위 큐에서 빼 줍니다.
- 마스크의 개수 X 만큼 사람들을 마스크에 매칭시킵니다. 즉, 우선순위 큐에서 사람들을 빼줍니다. 이 때 카운트를 해 줍니다.
- 이 과정을 반복하면 전체 마스크를 보고 나서 카운트된 값이 최대 매칭할 수 있는 사람들의 수가 됩니다.

1H. 전설

`hashing`, `trie`
출제진 의도 – **Medium**

- 제출 74번, 정답 4팀 (정답률 5.41%)
- 처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 33분
- 출제자: `shiftpsh`

1H. 전설

적절히 쉽게 풀리는 문제를 의도했습니다. 해싱 혹은 트라이로 풀 수 있습니다.

트라이 풀이를 소개합니다.

트라이 A에는 색상들을 집어넣습니다. 트라이 B에는 닉네임들을 **뒤집어서** 집어넣습니다.

쿼리가 들어올 때마다

- 쿼리로 들어온 문자열의 prefix 중 A에 들어 있는 문자열들의 길이들과
- 쿼리로 들어온 문자열을 **뒤집은 것**의 prefix 중 B에 들어 있는 문자열들의 길이

를 계산합니다. 이는 쿼리의 길이에 비례한 연산 횟수가 필요합니다.

전처리된 길이들의 집합을 각각 S_A, S_B 라고 합시다. 이 때 적절한 $s_A \in S_A, s_B \in S_B$ 가 있어서 $s_A + s_B$ 가 쿼리 문자열의 길이가 된다면 그 팀은 ICPC에서 수상할 수 있게 됩니다.

쿼리 문자열의 길이가 그렇게 길지 않으므로, S_A 와 S_B 를 `bool` 배열 등을 이용해 관리할 경우 빠른 시간 안에 판단할 수 있습니다. 또는, C++이라면 `std::bitset`을 이용해 $S_A \wedge S_B = \emptyset$ 인지를 판단할 수도 있습니다.

1J/2J. 상품권 준비

`greedy`, `prefix_sum`

출제진 의도 – **Medium**[Div 1] / **Challenging**[Div 2]

D1 제출 20번, 정답 4팀 (정답률 20.00%)

처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 17분

D2 제출 0번, 정답 0팀 (정답률 0.00%)

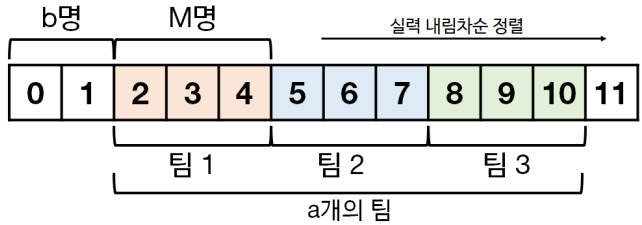
처음 푼 팀: —

– 출제자: `Green55`

1J/2J. 상품권 준비

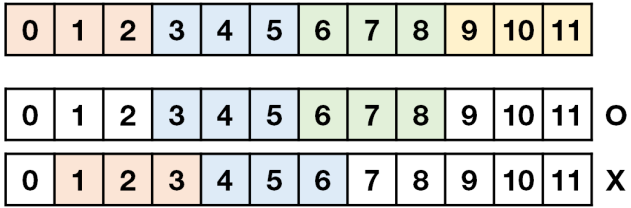
- 먼저 “최강의 팀 구성”을 만드는 법을 생각해봅시다.
- 최대한 실력이 높은 팀끼리 묶어주는게 이득입니다. (ex : $5 \times 4 + 3 \times 2 > 5 \times 3 + 4 \times 2$)
- 즉, 실력의 내림차순으로 정렬 후 앞에서부터 순서대로 묶어주는 것이 “최강의 팀 구성”입니다.

1J/2J. 상품권 준비



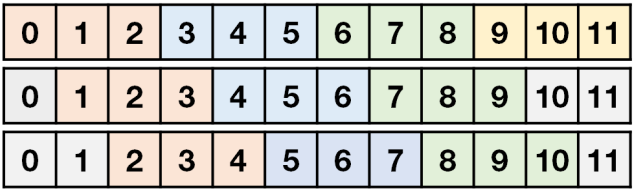
- 그렇다면 실력의 내림차순으로 이름을 써놓은 배열 $name[]$ 을 만들고
- a 와 b 가 주어지면 다음과 같은 값을 구하는 문제가 됐습니다. (0-based index)
- $\sum_{i=0}^{a-1} name[b+iM] \oplus name[b+iM+1] \oplus \cdots \oplus name[b+iM+M-1]$
- 쉽게 말해서 b 부터 시작해서 M 개씩 총 a 번 묶어주면 됩니다.

1J/2J. 상품권 준비



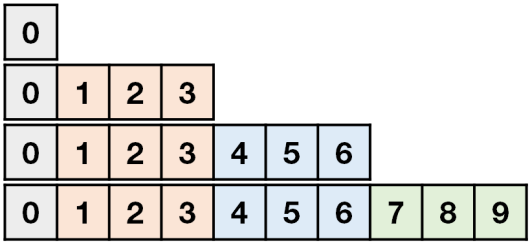
- 전처리를 해둬서 빠르게 쿼리에 답 하고 싶습니다. 그런데 b가 바뀌는게 짜증납니다.
- 예를 들어, $(name[0] \oplus name[1] \oplus name[2])$, $(name[3] \oplus name[4] \oplus name[5]), \ldots$ 를 한 덩어리로 묶어 누적합 전처리 해놨다면,
- $b = 0, 3, 6, \ldots$ 일 때 답을 $\mathcal{O}(1)$에 구할 수 있습니다. 하지만 다른 b일때는 어떡하죠?

1J/2J. 상품권 준비

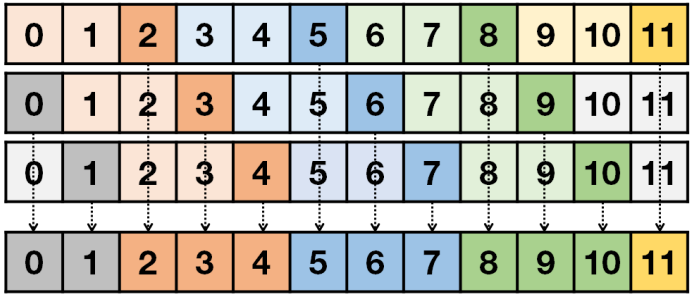


- 시작점이 $0, 1, \ldots, M-1$ 일 때의 모든 누적합 테이블을 만들어버리면 안될까요?
- 총 M 개의 테이블이 필요한데, 하나의 누적합 테이블을 만드는데 드는 시간은 $\mathcal{O}(N)$
- 곱하면 $\mathcal{O}(NM)$ 이니까 시간내에 만들 수 없겠군요.
- 과연 그럴까요?

1J/2J. 상품권 준비



- $x \oplus x = 0$ 입니다. 따라서 "누적 xor 합" 배열을 만들어 놓는 등의 방법으로, 추가할 "구간 xor 합"을 $\mathcal{O}(1)$에 구할 수 있습니다.
- 이렇게 하나의 누적합 테이블을 $\mathcal{O}(N/M)$에 채울 수 있고
- 그러면 M개의 테이블을 만드는 총 시간은 $\mathcal{O}(N)$입니다.



- 실제 구현은 누적합 테이블들을 1차원에 전부 합치면 간단합니다.
- 앞에서부터 채우면, 매번 필요한 "구간 xor 합"을 슬라이딩 윈도우로 구할 수 있습니다.

1F. 울타리

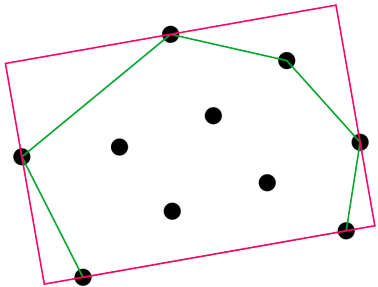
`geometry`, `convex_hull`, `rotating_calipers`

출제진 의도 – **Hard**

- 제출 11번, 정답 1팀 (정답률 9.09%)
- 처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 103분
- 출제자: `semteo04`

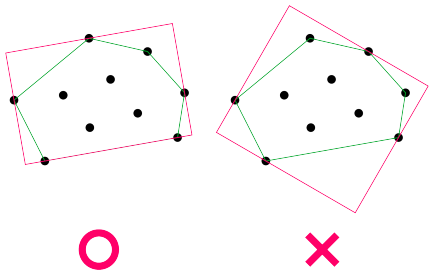
1F. 울타리

점들이 주어지면 해당 점들을 포함하는 가장 작은 컨벡스 헐을 만듭니다.
이 컨벡스 헐에 외접하는 직사각형들 중 둘레가 가장 작은 직사각형을 구해야 합니다.



1F. 울타리

컨벡스 헐에 외접하는 직사각형들 중 컨벡스 헐의 한 변을 포함하는 것들만 생각합니다. 그렇게 고른 직사각형들 중 정답이 있음이 보장됩니다.
해당 내용의 증명은 위키백과[1]에서 확인할 수 있습니다.



[1] https://en.wikipedia.org/wiki/Minimum_bounding_box

1F. 울타리

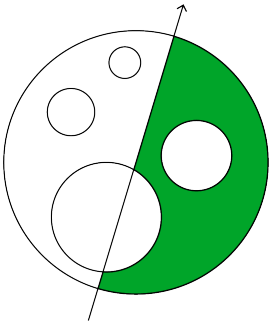
로테이팅 캘리퍼스를 90° 간격으로 두고 한 변을 포함할 때까지 돌립니다. 이 동작을 계속 반복하면서 둘레의 길이를 구하고, 최소가 되는 값을 출력합니다.

1C. 케이크 커팅

`geometry`
출제진 의도 – **Hard**

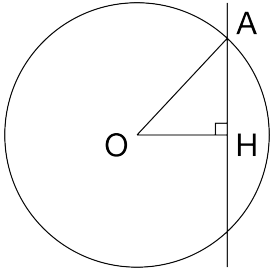
- 제출 2번, 정답 1팀 (정답률 50.00%)
- 처음 푼 팀: **햇빛이 선명하게 문제를 핥고 있었다**[연세대학교] (윤인섭, 황준호, 남현우), 175분
- 출제자: `semteo04`

- x 축과 칼이 이루는 각도를 θ 라고 하고, $f(\theta) = (\text{칼 오른쪽에 있는 케이크의 넓이})$ 라고 합시다.

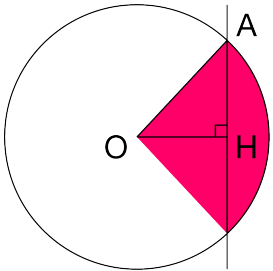


1C. 케이크 커팅

- 원의 중심 좌표를 (x, y), 칼에 해당하는 일차함수를 $ax + by + c = 0$이라 합시다.
- 원의 중심을 O, O에서 칼쪽으로 수선의 발을 내린 점을 H라 합시다.
- $\overline{OH} = \dfrac{|ax + by + c|}{\sqrt{a^2 + b^2}}$

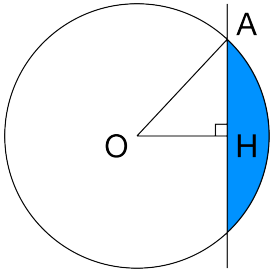


- $\cos(\angle AOH) = \dfrac{\overline{OA}}{\overline{OH}}$ 이므로 $\arccos$ 을 이용해 $\angle AOH$ 를 구할 수 있습니다.
- 이것을 이용해서 빨간 부분의 넓이 S 를 구할 수 있습니다.



1C. 케이크 커팅

- 피타고라스 정리를 이용해 $\overline{AH}$ 를 구할 수 있고
- $S - \overline{AH} \times \overline{OH}$ 로 파란 부분의 넓이를 구할 수 있습니다.



이를 모든 페퍼민트에 대해 계산해 적절히 빼고 더하면, θ 에 대한 $f(\theta)$ 를 구할 수 있습니다.

케이크 전체 넓이를 A라고 할 때,

- $f(0) + f(\pi) = A$이고
- $f(\theta)$는 연속함수

이므로 $f(0)$과 $f(\pi)$ 안에 $A/2$인 지점이 존재합니다.

따라서 이분 탐색으로 해결 가능합니다.

$s = 0, e = \pi$로 두고, 편의상 $f(s) >= A/2, f(e) <= A/2$라고 가정한다면

- $f((s+e)/2) >= A/2$라면 $[(s+e)/2, e]$에 $f(\theta) = A/2$인 θ가 있고
- $f((s+e)/2) <= A/2$라면 $[s, (s+e)/2]$에 $f(\theta) = A/2$인 θ가 있습니다.

1K. 물건 가져가기

`flow`, `mfmc`

출제진 의도 – **Hard**

- 제출 14번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: —
- 출제자: `lky7674`

1K. 물건 가져가기

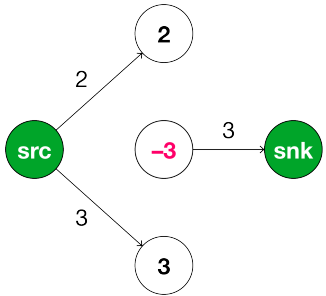
- 출제진의 의도는 3–2 정도였지만 검수진들 의견에 따라서 Diamond로 승격한 문제입니다...
- SCC로 풀릴 수 있지 않을까? 생각한 당신! 유감입니다. 출제진/검수진 중 아무도 SCC로 풀이를 성공하지 못했습니다.
- 보기와는 다르게 min cut으로 풀리는 문제입니다.
- Dinic이 아닌 Edmond-Karp로도 0ms로 넉넉하게 통과하도록 만들었습니다.

1K. 물건 가져가기

처음에는 source, sink 노드와 각 물건들에 대한 정점들을 만들어줍니다. 각 물건들의 대한 정점들을 v_i 로 정의합니다. 다음 그림은 2번째 예제에 대한 그림입니다.

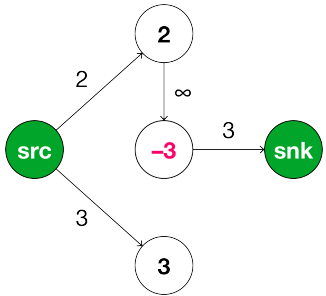
1K. 물건 가져가기

$t_i > 0$ 이면 t_i 의 capacity를 가지고 source에서 v_i 로 가는 간선을 생성
$t_i < 0$ 이면 $-t_i$ 의 capacity를 가지고 v_i 에서 sink로 가는 간선을 생성
$t_i = 0$ 이면 둘 중 아무거나 선택해도 됩니다.



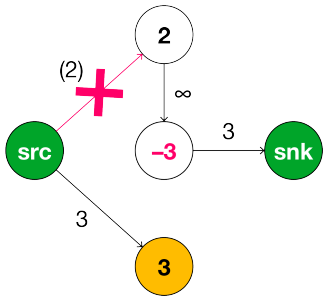
1K. 물건 가져가기

∞의 capacity를 가지고 s_i에서 e_i로 가는 간선을 생성합니다.



1K. 물건 가져가기

해당 그래프에서 min cut을 구하고 source와 같은 쪽에 있는 물건들만 고르면 정답이 됩니다.



1K. 물건 가져가기

간단한 증명:

- C를 $\sum_{t_i \geq 0} t_i$으로 정의합니다.
- $\sum_{i \in A} t_i$의 최댓값을 구하는 것은 $C - \sum_{i \in A} t_i$를 최소화하는 것과 같습니다.
- $C - \sum_{i \in A} t_i = \sum_{i \in A, t_i < 0} (-t_i) + C - \sum_{i \in A, t_i \geq 0} t_i$로 A가 제약조건들을 전부 만족한다면 우변은 해당 그래프에서의 $c(\{\text{source}\} \cup A, \{\text{sink}\} \cup (V - A))$의 cut capacity입니다.
- 해당 cut capacity를 최소화해서 얻은 A는 $\sum_{i \in A} t_i$의 최댓값을 얻어냅니다.

1D. 가뭄 (Large)

`linear_programming`, `duality`, `mcmf`

출제진 의도 – **Challenging**

- 제출 11번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: –
- 출제자: `lky7674`

1D. 가뭄 (Large)

풀이 요약:

- LP를 푸는 알고리즘인 simplex algorithm은 시간 초과를 받습니다.
- LP를 dual LP로 변경한 뒤에 적절한 cost가 주어진 flow를 만들어 주면 됩니다.

다음과 같은 linear programming 이 주어진다고 생각해 봅시다.

$$\text{maximize} \sum_{i=1}^{m} c_i x_i$$

$$\text{subject to} \sum_{i=1}^{m} a_{ij} x_i \leq b_j \qquad \text{for } j = 1, \cdots, n$$

$$x_i \geq 0 \qquad \text{for } i = 1, \cdots, m$$

이 때의 dual LP는 다음과 같습니다.

$$\text{minimize} \sum_{j=1}^{n} b_j y_j$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} y_j \geq c_i \qquad \text{for } i = 1, \cdots, m$$

$$y_j \geq 0 \qquad \text{for } j = 1, \cdots, n$$

1D. 가뭄 (Large)

– 일반화된 LP들을 dual LP로 변경하는 방법은 다음과 같습니다.

Primal maximization	Dual minimization
$\leq b_i$	$y_j \geq 0$
$\geq b_i$	$y_j \leq 0$
$= b_i$	y_j : free
$x_i \leq 0$	$\leq c_j$
$x_i \geq 0$	$\geq c_i$
x_i : free	$= c_j$

두 LP가 있을 때, 2가지 theorem을 만족하게 됩니다.

- Theorem (Weakly duality) : x가 LP의 feasible solution이고, y가 dual LP의 feasible solution이면 $\sum_{i=1}^{m} c_i x_i \leq \sum_{j=1}^{n} b_j y_j$를 만족한다.
- Theorem (Strong duality) : x^*가 LP의 optimal solution이고, y^*가 dual LP의 optimal solution이면 $\sum_{i=1}^{m} c_i x_i^* = \sum_{j=1}^{n} b_j y_j^*$를 만족한다.

문제에 맞는 LP는 다음과 같습니다.

$$\text{maximize} \sum_{i=1}^{n}(a_i - b_i)$$

$$\text{subject to } a_i - b_j \leq c_{i,j} \quad \text{for } i, j = 1, \cdots, n$$

$$a_i \geq 0 \quad \text{for } i = 1, \cdots, n$$

$$b_i \geq 0 \quad \text{for } i = 1, \cdots, n$$

1D. 가뭄 (Large)

a_i, b_i 는 free variable로 변경해도 $\max$ 값은 변하지 않기 때문에 바꿔도 됩니다.

$$\text{maximize} \sum_{i=1}^{n}(a_i - b_i)$$

$$\text{subject to } a_i - b_j \leq c_{i,j} \qquad \text{for } i, j = 1, \cdots, n$$

$$a_i, b_i\text{: free} \qquad \text{for } i = 1, \cdots, n$$

이를 dual LP로 바꾸면 다음과 같습니다.

$$\text{minimize} \sum_{i,j=1,\dots,n} c_{i,j}x_{i,j}$$

$$\text{subject to} \sum_{i=1}^{n} x_{i,j} = 1 \qquad \text{for } j = 1, \cdots, n$$

$$\sum_{j=1}^{n} -x_{i,j} = -1 \qquad \text{for } i = 1, \cdots, n$$

$$x_{i,j} \geq 0 \qquad \text{for } i, j = 1, \cdots, n$$

1D. 가뭄 (Large)

부호를 변경하면...

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,j=1,\dots,n} c_{i,j} x_{i,j} \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{i,j} = 1 && \text{for } j = 1, \cdots, n \\
& \sum_{j=1}^{n} x_{i,j} = 1 && \text{for } i = 1, \cdots, n \\
& x_{i,j} \geq 0 && \text{for } i, j = 1, \cdots, n
\end{aligned}$$

...**최소 가중치 이분 매칭**과 같은 문제가 됩니다. 여기서부터는 MCMF 문제와 같은 방법으로 해결할 수 있습니다.

1L. 카드 셔플

`splay_tree`
출제진 의도 – **Challenging**

- 제출 14번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: –
- 출제자: `semteo04`

1L. 카드 셔플

- 스플레이 트리라는 자료 구조를 사용합니다.
- 스플레이 트리는 서브트리를 다른 곳으로 이동시키는데 $\mathcal{O}(\log N)$ 이 걸립니다.

1L. 카드 셔플

- 리프 노드의 개수가 N 개인 이진 트리를 만들고 리프 노드의 왼쪽부터 $1, 2, \ldots n$ 을 채워 넣습니다.
- 1, 2번 쿼리 : 스플레이 트리를 이용하면 해당 구간을 $\mathcal{O}(N \log N)$ 에 해결 가능합니다.

- 3번 쿼리 : 해당 구간에 해당하는 정점들을 DFS로 순회하면서 배열에 저장합니다. 이 동작에 $\mathcal{O}(1000 + \log N)$ 만큼의 시간이 듭니다. 셔플을 하고 다시 순회하면서 셔플한 카드들을 넣어줍니다.

1B. 꿀벌

`dp_connection_profile`
출제진 의도 – **Challenging**

- 제출 0번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: —
- 출제자: `shiftpsh`

1B. 꿀벌

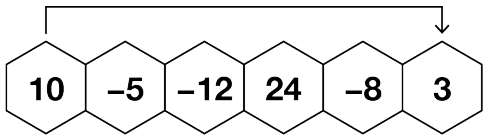
일단 벌의 동작을 살펴봅시다.

- 인접한 칸으로 움직인다.
- 날아간다.

일단 '날아가는 동작'에 대해 먼저 이해해 보도록 합시다. 어떤 상황에서 인접한 칸으로 움직이는 게 이득이고, 어떤 상황에서 날아가는 게 이득일까요?

1B. 꿀벌

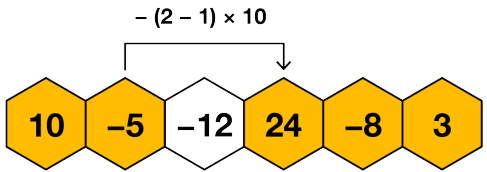
이차원 벌집은 생각하기 어려우니 일차원 벌집에서 생각해 봅시다. 예를 들어 아래와 같은 상황에서 $F = 10$ 이고, 첫 칸에서 시작해 끝 칸에 도달하고 싶다고 생각해 봅시다.



'지나온 경로를 다시 지날 때는 에너지를 소모하지 않는다' 는 조건이 있긴 하지만 다행히도 이런 상황에서는 무시할 수 있습니다.

1B. 꿀벌

소모하는 에너지가 $((\text{벌집 거리}) - 1) \times F$ 라고 했는데, 두 칸의 $(\text{벌집 거리}) - 1$ 은 두 칸 사이의 칸들의 수와 같습니다.



따라서 이렇게 벌집이 일자인 경우에는 벌집 안의 수가 $-F$ 보다 작은 경우만 날아서 지나가면 됩니다.

따라서 일차원 벌집의 경우에는

- $-F$ 보다 작은 값의 칸의 값을 모두 $-F$ 로 교체하고
- 모든 연속 구간들 중에서, 그 구간 내 원소들의 합이 최댓값이 되는 경우가 언제인지를 구하면 되는

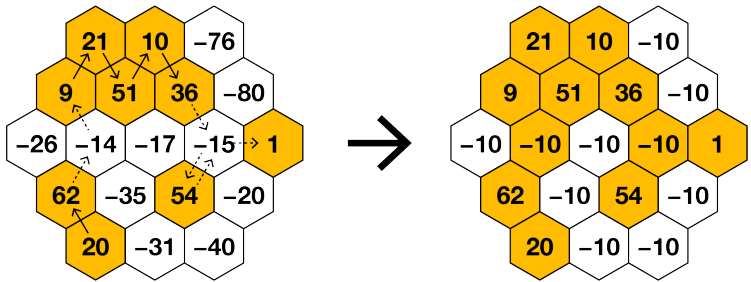
쉬운 문제가 됩니다. 최대 구간합을 구하는 알고리즘은 잘 알려져 있습니다.

위에서 우리는 날아갈 때 소모되는 에너지를 아예 칸의 값으로 대체해 버렸는데, 아주 고맙게도 **'한 번 이상 지나온 경로를 다시 지날 때는 에너지가 소모되지 않는다'**는 조건 덕분에, 이차원에서도 $-F$ 보다 작은 값의 칸의 값을 모두 $-F$ 로 교체하는 아이디어를 적용할 수 있습니다.

– 잘 생각해 보면 '한 번 지나온 경로'는 '한 번 지나온 칸'에 대한 이야기로 바꿀 수 있습니다.

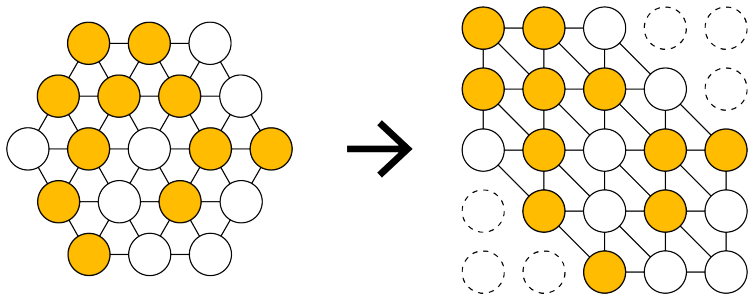
1B. 꿀벌

이제 벌집 내의 연결 요소 중 합이 가장 큰 것을 고르는 문제가 되었습니다. 물론 Kadane은 쓸 수 없습니다.



1B. 꿀벌

육각형은 생각하기 힘드니 행렬로 바꿔서 생각해봅시다.

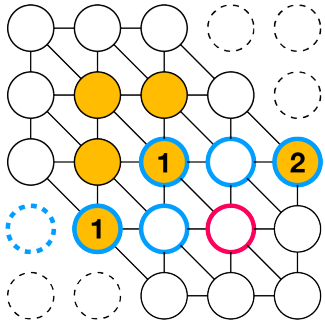


이 행렬을 A라 하고, 각 칸의 값을 A_{ij}라고 합시다. 행렬의 크기는 $(2N-1) \times (2N-1)$인데, 편의상 $M = 2N-1$이라고 합시다.

여기서 다이나믹 프로그래밍을 생각해볼 수 있습니다. $dp\,(i, j, f, s)$ 를 아래와 같이 정의합시다.

- A_{ij} 를 $f = 0$ 이면 선택하지 않고, $f = 1$ 이면 선택했을 때,
- 이전 $M + 1$ 개 칸 $(A_{(i-1)j}, A_{(i-1)(j+1)}, \cdots, A_{(i-1)(M-1)}, A_{i0}, A_{i1}, \cdots, A_{i(j-1)})$ 의 **선택 상태**가 s 인 경우,
- A_{00} 부터 A_{ij} 까지의 $iM + j + 1$ 개 칸 중에서 연결 요소를 골랐을 때 얻을 수 있는 에너지의 최댓값

1B. 꿀벌



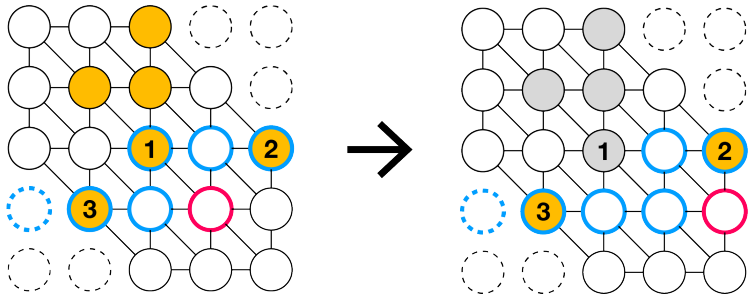
선택 상태 s 는 파란색 테두리의 노드들이 어느 연결 요소에 들어 있는지를 알려줍니다. 위 그림과 같은 상황에서는 102010 으로 표현할 수 있습니다. 비트필드 DP와 유사하죠.

1B. 꿀벌

우리는 선택한 칸들이 연결 요소를 이루는지에만 관심이 있습니다. 다른 말로는 현재 칸을 선택하지 않았을 때 연결 요소가 무조건 2개 이상이 되어버린다면 $-\infty$를 저장하는 등의 방식으로 그냥 상태 전이를 하지 않아버리면 됩니다.

이는 최근 $M+1$개 칸을 보는 것만으로 충분합니다. 현재 칸을 선택하지 않았을 때, 현재 칸의 왼쪽 위 칸이 고립되는 경우만 고려해 주면 되기 때문입니다. 여기서 고립되는지 아닌지를 판단하기 위해 연결 요소의 번호가 필요하며, 단순히 이전 칸들이 선택되었는지 아닌지를 저장하는 비트필드 DP의 방식으로는 해결하기 힘듭니다.

1B. 꿀벌



왼쪽 상태에서 현재 칸을 선택하지 않는 경우 1번 연결 요소는 고립되어 버리고, 이후 어떻게 선택하더라도 하나의 연결 요소가 되지 않습니다.
이 상황만 고려해도 연결 요소가 유지되는지 판단하는 데에는 충분합니다.

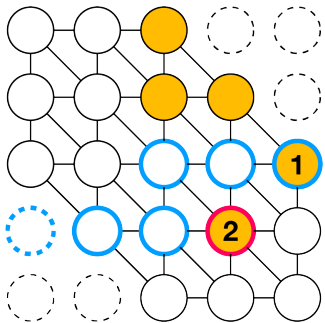
반대로 왼쪽 위의 칸이 비어 있었다면 현재 칸을 선택하든 선택하지 않든 연결 요소 유지와는 관련이 없습니다. 따라서 현재 칸을 선택하지 않는 경우는 s의 첫 번째 수가 0인지 아닌지만 고려해 주면 됩니다.

또한 현재 칸을 선택하지 않음으로써 연결 요소가 하나 결정되는 순간 그 칸 이하에는 상태 전이가 일어나지 않기 때문에, 칸을 선택하는 건 걱정 없이 할 수 있습니다. 쉽게 말하면 칸을 선택할 때는 연결 요소가 위에서 '끊겨' 있는 경우는 고려할 필요가 없습니다.

그러면 현재 칸을 선택하는 경우를 생각해 봅시다. 이 때는 상태 전이를 할 때 현재 칸의 왼쪽 칸, 왼쪽 윗 칸, 윗 칸의 세 칸을 고려해 줘야 합니다.

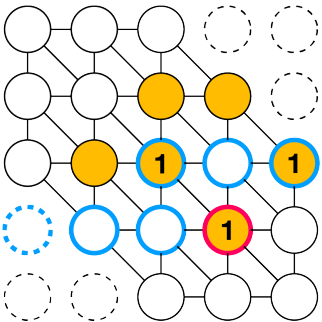
1B. 꿀벌

일단은 세 칸이 모두 비어 있는 경우입니다. 이 경우에는 s에 존재하지 않는 새 연결 요소 번호를 배정받습니다.



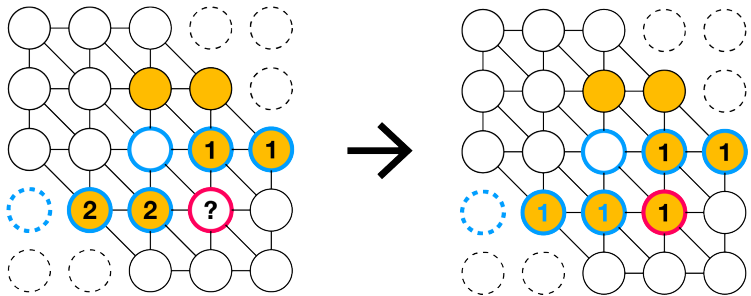
1B. 꿀벌

세 칸에 배정되어 있는 0이 아닌 번호가 한 개뿐인 경우, 그 번호를 따라갑니다.



1B. 꿀벌

세 칸에 배정되어 있는 0이 아닌 번호가 두 개인 경우, 두 번호를 합쳐 줘야 합니다. 아래의 경우엔 1과 2가 하나의 연결 요소를 이뤘으므로 2를 1로 바꿔 주었습니다.



이렇게 DP 테이블을 채워나가면서, i, j, f 와 상관없이 s에 저장되어 있는 0이 아닌 수의 종류가 1개인 경우(즉, 연결 요소가 하나로만 결정되는 경우)들만 모아서 그 중의 최댓값을 출력하면 됩니다.

그러나 s를 관리하는 데에 있어서도 약간의 생각이 필요합니다. $N = 6$인 경우 s는 12글자가 되는데, 이를 그냥 배열 인덱스로 쓰기에는 공간과 시간 모두가 문제입니다.

존재할 수 있는 s의 종류 자체가 얼마 안 되기 때문에 sparse하게 관리하는 것이 유리합니다. `map` 등의 자료 구조를 이용해 s들을 관리할 수 있습니다. 또한, 예를 들어, 105044와 102033은 같은 상태라는 것을 알 수 있습니다. 이런 경우들을 잘 최적화해준다면 s로 가능한 상태의 수를 훨씬 줄일 수 있습니다.

이런 최적화를 거치면 200ms – 400ms 정도에 동작하는 풀이를 작성할 수 있습니다.