SUAPC 2021w 풀이

Official Solutions

신촌지역 대학생 프로그래밍 대회 동아리 연합

2021년 2월 28일

240

문제		의도한 난이도	출제자
A	우선순위 계산기	Hard	이국렬 lky7674
B	떡국	Easy	이상원 gumgood
C	반짝반짝	Medium	박수현 shiftpsh
D	달고나	Hard	이상원 gumgood
E	시철이가 사랑한 수식	Challenging	정기웅 QuqqU
F	성싶당	Medium	윤지학 cubelover, 박수현 shiftpsh
G	신촌지역 초중고등학생 프로그래밍 대회...	Hard	정연두 Green55
H	카카오톡	Easy	이국렬 lky7674
I	팰린드롬 척화비	Easy	정기웅 QuqqU
J	의자 게임	Hard	정연두 Green55
K	합성인수분해	Medium	정연두 Green55
L	습격받은 도시	Medium	이상원 gumgood
M	Go와 함께하는 전화망 서비스	Challenging	이국렬 lky7674

A. 우선순위 계산기

240

parsing, priority_queue, linked_list, tree$_{set}$
출제진 의도 – **Hard**

- 제출 83번, 정답 4팀 (정답률 4.819%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 53분
- 출제자 : 이국렬$^{\text{lky7674}}$

240

- 독특한 계산기, 뒤집힌 계산기의 뒤를 이어서 뇌절까지 해서 나온 계산기 문제입니다.
- 다만 위의 두 문제와 다르게 구현이 상당히 까다롭습니다.

240

- 우선 문자열을 Parsing 해야 합니다.
- 문자열을 Parsing 할 때, 숫자를 읽으면 (지금까지 읽은 수) × 10 + (현재 읽고 있는 숫자) 로 지금까지 읽은 숫자를 갱신.
- 연산자를 읽으면 지금까지 읽은 수와 연산자를 배열에 추가합니다.

240

- 이웃한 숫자들을 계산한 값과 연산자 우선순위에 대한 구조체를 Set이나 Priority Queue같은 자료구조에 저장합니다.
- 먼저 계산해야 할 것을 뽑고, 이웃한 연산자에 대한 계산값을 갱신하고 다시 자료구조를 갱신합니다.
- 정해는 Priority Queue랑 Linked List로 정리하는 것이지만, Set과 Map을 사용해도 통과가 되도록 만들었습니다.

240

B. 떡국

greedy
출제진 의도 – **Easy**

- 제출 185번, 정답 44팀 (정답률 23.784%)
- 처음 푼 팀 : **3M** (서강대학교), 3분
- 출제자 : 이상원gumgood

240

- 두 떡국 그릇에 대해 다음과 같은 사실을 관찰할 수 있습니다.
  - 크기가 다른 경우, 하나의 **떡국 그릇 탑**에 속할 수 있습니다.
  - 크기가 같은 경우, 하나의 **떡국 그릇 탑**에 속할 수 없습니다.
- 이로부터 크기가 i 인 떡국 그릇이 c_i 개가 있을 때,
  최소 c_i 개의 **떡국 그릇 탑**이 있어야 한다는 사실을 알 수 있습니다.

240

- 따라서 모든 떡국 그릇은 적어도 $\max c_i$ 개 이상의 **떡국 그릇 탑**이 됩니다.
- 또한 정확히 $\max c_i$ 개의 **떡국 그릇 탑**으로 만들 수 있습니다.
  - 크기가 i 인 그릇을 포함하지 않는 **떡국 그릇 탑**의 개수가 항상 c_i 개 이상 존재합니다.

240

- 정리하면, 크기별로 떡국 그릇의 개수를 세었을 때 가장 큰 값이 답입니다.
- 서로 다른 크기의 떡국 그릇 수 C에 대해
  $\mathcal{O}(N \log N)$, $\mathcal{O}(N + C)$ 등의 다양한 방법으로 해결할 수 있습니다.

C. 반짝반짝

dp, probability

출제진 의도 – **Medium**

- 제출 54번, 정답 8팀 (정답률 14.815%)
- 처음 푼 팀 : **팬케이크맛쿠키** (연세대학교), 52분
- 출제자 : 박수현 shiftpsh

전구 스트립이 하나 있다고 생각해 봅시다. 이 때 기댓값을 어떻게 계산할 수 있을까요?

구체적으로, 전구 스트립 P가 있어서, 각각의 전구가 고장날 확률이 순서대로 $p_0, p_1, \cdots, p_{N-1}$이라고 해 봅시다.

- 첫 번째 전구가 고장나면, 켜진 전구의 수는 0 이고, 이런 상황이 일어날 확률은 p_0 입니다.
- 첫 번째까지의 전구가 멀쩡하고 두 번째 전구가 고장나면, 켜진 전구의 수는 1 이고, 이런 상황이 일어날 확률은 $(1 - p_0)\, p_1$ 입니다.
- 두 번째까지의 전구가 멀쩡하고 세 번째 전구가 고장나면, 켜진 전구의 수는 2 이고, 이런 상황이 일어날 확률은 $(1 - p_0)\,(1 - p_1)\, p_2$ 입니다.
- $\cdots$ N 개의 전구가 다 켜져 있을 확률은 $\displaystyle\prod_{i=0}^{N-1} (1 - p_i)$ 입니다.

$\cdots$ 따라서, k개의 전구가 켜져 있을 확률, 즉 $\Pr(X = k)$는

$$\Pr(X = k) = \begin{cases} p_k \prod_{i=0}^{k-1} (1 - p_i) & k < N \\ \prod_{i=0}^{N-1} (1 - p_i) & k = N \end{cases}$$

가 되고, 전구 스트립 P에 대한 기댓값 $E(P)$는 정리하면

$$E(P) = \sum_{k=0}^{N} k \Pr(X = k) = \sum_{k=1}^{N-1} \left[k p_k \prod_{i=0}^{k-1} (1 - p_i) \right] + N \prod_{i=0}^{N-1} (1 - p_i)$$

가 됩니다.

이제 큰 스트립을 작은 스트립으로 어떻게 쪼갤 수 있을지를 생각해 봅시다. 일단 이전 슬라이드에서와 같은 방법으로 P의 어떤 구간 $[l, r]$에서의 기댓값은 아래와 같은 식으로 계산할 수 있습니다.

$$E\left(P_{[l,r]}\right) = \sum_{k\in[l,r]} \left[(k-l)\, p_k \prod_{i=l}^{k-1} (1-p_i) \right] + (r-l+1) \prod_{i=l}^{r} (1-p_i)$$

이 식을 계산할 경우, 한 구간에 대해 $\mathcal{O}\left(N^2\right)$ 번, 모든 구간에 대해서는 $\mathcal{O}\left(N^4\right)$ 번의 계산이 필요합니다. 줄일 수 있을까요?

$$E\left(P_{[l,r]}\right) = \sum_{k\in[l,r]} \left[(k-l)\, p_k \underbrace{\prod_{i=l}^{k-1} (1-p_i)}_{\triangleq c_{[l,k-1]}} \right] + (r-l+1) \underbrace{\prod_{i=l}^{r} (1-p_i)}_{\triangleq c_{[l,r]}}$$

c를 위 식에서의 파란색 부분과 같이 정의합시다. 그러면 임의의 구간에 대해 $c_{[l,r)}$ 은 아래와 같은 점화식으로 계산할 수 있습니다.

$$c_{[l,r]} = \prod_{i\in[l,r]} (1-p_i) = \begin{cases} (1-p_r)\, c_{[l,r-1]} & l<r \\ 1-p_l = 1-p_r & l=r \end{cases}$$

240

$$c_{[l,r]} = \prod_{i\in[l,r]} (1 - p_i) = \begin{cases} (1 - p_r)\, c_{[l,r-1]} & l < r \\ 1 - p_l = 1 - p_r & l = r \end{cases}$$

이 점화식을 이용해 다이나믹 프로그래밍하면, 모든 $[l, r]$에 대해 $c_{[l,r]}$을 $\mathcal{O}\left(N^2\right)$로 전처리해둘 수 있습니다.

- prefix sum 등을 사용할 수도 있습니다.

c를 도입하면 기존의 식을 아래와 같이 바꿀 수 있습니다.

$$\begin{aligned} E\left(P_{[l,r]}\right) &= \sum_{k\in[l,r]} \left[(k-l)\, p_k \prod_{i=l}^{k-1} (1-p_i)\right] + (r-l+1) \prod_{i=l}^{r} (1-p_i) \\ &= \sum_{k\in[l,r]} \left[(k-l)\, p_k c_{[l,k-1]}\right] + (r-l+1)\, c_{[l,r]} \end{aligned}$$

전처리한 c를 사용하면, 한 구간에 대해 $\mathcal{O}(N)$ 번의 계산만으로 기댓값을 구할 수 있습니다.
모든 구간에 대해서는 $\mathcal{O}(N^3)$ 입니다.

기댓값의 식의 형태를 유심히 관찰하면 $E\left(P_{[l,r]}\right)$ 도 점화식으로 표현하는 게 가능하다는 사실을 알 수 있습니다. 편의를 위해 $l < r$ 이라고 하면, 아래와 같이 됩니다.

$$\begin{aligned}
E\left(P_{[l,r]}\right) &= \sum_{k\in[l,r]} \left[(k-l)\,p_k c_{[l,k-1]}\right] + (r-l+1)\,c_{[l,r]} \\
&= \sum_{k\in[l,r-1]} \left[(k-l)\,p_k c_{[l,k-1]}\right] + (r-l)\,p_r c_{[l,r-1]} + (r-l+1)\,c_{[l,r]} \\
&= E\left(P_{[l,r-1]}\right) - (r-l)\,c_{[l,r-1]} + (r-l)\,p_r c_{[l,r-1]} + (r-l+1)\,c_{[l,r]} \\
&= E\left(P_{[l,r-1]}\right) - (r-l)\,(1-p_r)\,c_{[l,r-1]} + (r-l+1)\,c_{[l,r]}
\end{aligned}$$

$l = r$ 일 때는 $E\left(P_{[l,l]}\right) = 1 - p_l$ 이 됩니다.

따라서 모든 구간의 기댓값을 다이나믹 프로그래밍으로 $\mathcal{O}\left(N^2\right)$ 에 전처리할 수 있습니다.

이제 다음과 같은 함수를 생각해 봅시다.

$$f\left(i,k\right) = \left(\text{구간 } [0,i] \text{ 를 } k\text{개의 구간으로 나눌 때의 최대 기댓값}\right)$$

240

그러면 $f\left(i,j\right)$는 다음과 같은 점화식으로 정의할 수 있습니다.

$$f\left(i,k\right)=\begin{cases}\max\limits_{m\in[1,i)}\left[f\left(m-1,k-1\right)+E\left(P_{[m,i]}\right)\right] & k>1\\ E\left(P_{[0,i]}\right) & k=1\end{cases}$$

답은 $\max\limits_{k\in[1,K)} f\left(N-1,k\right)$가 되고, $\mathcal{O}\left(N^2K\right)$만에 계산 가능합니다. 이는 문제를 해결하기에 충분합니다.

D. 달고나

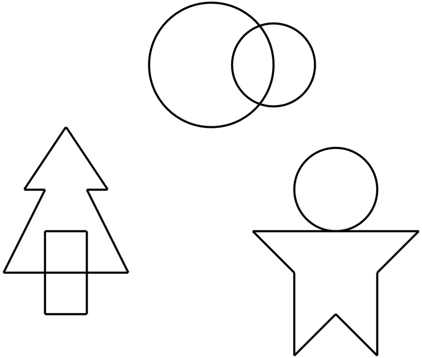
euler_characteristic, geometry, graph

출제진 의도 – **Hard**

- 제출 8번, 정답 0팀 (정답률 0.000%)
- 처음 푼 팀 : — (—), —분
- 출제자 : 이상원gumgood

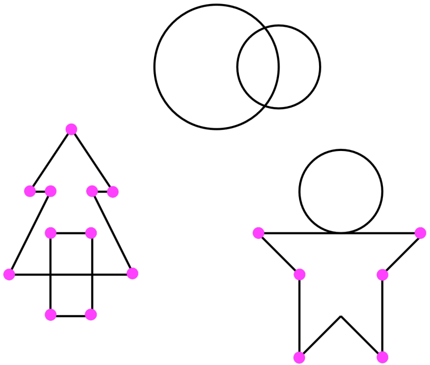
Euler's Polyhedral Formula : 연결된 평면 그래프에서 정점의 개수가 V 개, 간선의 개수가 E 개, 나눠지는 영역의 개수가 F 일 때, $V - E + F = 2$ 가 성립한다.

240

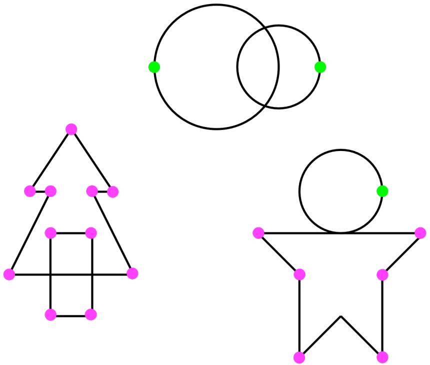


✓ 입력으로 주어지는 원 또는 단순 다각형들을 연결된 평면 그래프로 만들어 봅시다.

240

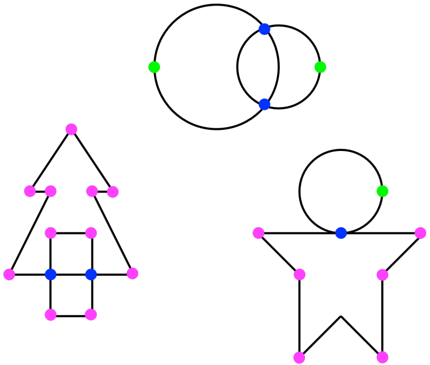


✓ 단순 다각형들의 꼭짓점에 정점을 놓습니다.



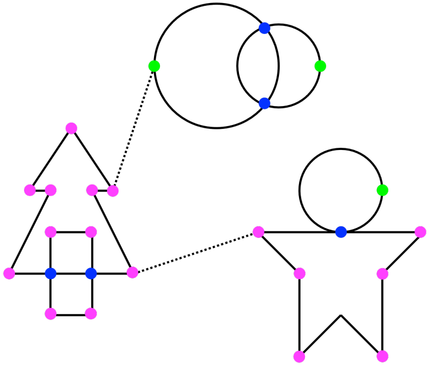
- 원 위의 임의의 위치에 정점을 하나 놓습니다.

240



✓ 모든 선들간의 교점을 정점을 놓습니다.

240



✓ 마지막으로 (컴포넌트 개수)-1개의 간선을 추가하여 모든 그래프를 연결합니다.

- 정점 개수는 다음과 같이 계산할 수 있습니다.
  - $V = \sum$ (다각형 꼭짓점 개수) $+$ (원의 개수) $+$ (교점 개수)
- 간선 개수는 다음과 같이 계산할 수 있습니다.
  - 세 선이 한 점을 지나는 경우는 없으므로
    교점이 하나 있을 때마다 간선이 두 개 늘어납니다.
  - $E = \sum$ (다각형 선분 개수) $+$ (원 개수) $+$ (교점 개수) $\times 2 +$ (컴포넌트 개수) $- 1$
- 영역 개수는 다음과 같이 계산할 수 있습니다.
  - $\sum$(다각형 꼭짓점 개수) $= \sum$(다각형 선분 개수)
  - $V - E + F = 2$
  - $F =$ (교점의 개수) $+$ (컴포넌트 개수) $+ 1$

240

- 정리하면, (교점의 개수) $+$ (컴포넌트 개수) $+\ 1$ 가 답입니다.
- 선분 및 원의 교차판정과 DFS/BFS를 통해 쉽게 해결할 수 있습니다.
  - 실수 자료형을 사용하는 경우 1e-13 보다 높은 정밀도로 구현해야 합니다.
- 전체 시간복잡도는 $\mathcal{O}\left(N^2\right)$ 입니다.

240

E. 시철이가 사랑한 수식

`number_theory, mobius_inversion`
출제진 의도 – **Challenging**

- 제출 33번, 정답 1팀 (정답률 3.030%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 270분
- 출제자 : 정기웅QuqqU

240

- $$\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)\times\operatorname{lcm}(i,j)$$
- $$\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)\times\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$$
- 시철이가 사랑하게 된 두 수식이 값이 어떤지 알아보면 됩니다.
- 수식 전개 과정이 **매우매우매우** 길어서, 집중력이 필요합니다.
- 결론부터 말하면 $\mathcal{O}(N)$ 만에 두 수식의 값을 구할 수 있습니다.

240

- 식 1의 값을 구하는 것은 상대적으로 쉽습니다.
- $\gcd(i, j) \times \mathrm{lcm}(i, j) = i \times j$ 인 것을 어렵지 않게 알 수 있고, 이것을 이용해 수식을 전개합니다.

240

$$\begin{aligned}\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)\times\operatorname{lcm}(i,j) &= \sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n} i\times j \\ &= \sum_{n=1}^{N}\sum_{i=1}^{n} i\times\frac{n(n+1)}{2} \\ &= \sum_{n=1}^{N}\frac{n(n+1)}{2}\times\frac{n(n+1)}{2}\end{aligned}$$

수식을 한번 더 전개해 $\mathcal{O}(1)$ 로 풀 수 있지만, 여기까지만 전개해도 $\mathcal{O}(N)$ 으로 충분히 AC가 가능합니다.

- 이제, 식 2의 값을 구하는 것이 문제입니다.
- $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{gcd}(i,j)$ 와 $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{lcm}(i,j)$ 의 값을 각각 $\mathcal{O}(N)$ 에 구해 곱해줘야 합니다.
- 먼저, $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{gcd}(i,j)$ 의 값을 구해 봅니다.

E. 시철이가 사랑한 수식

$\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$ 의 값은 4단계에 걸쳐 구할 수 있습니다.

1. $\sum_{d|n}\mu(d) = [n = 1]$ (Mőbius function 의 성질)
2. $\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j) = 1]$ (서로소 쌍의 개수)
3. $\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$ (모든 쌍의 $\gcd$ 합)
4. $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$

1. $\sum_{d|n} \mu(d) = [n = 1]$ (Mőbius function의 성질)

- Mőbius function 의 널리 알려진 성질입니다. 정의보다 이 성질이 더 유용합니다.
- $[condition]$ 기호는 안의 $condition$ 이 참이면 1, 거짓이면 0 의 값을 나타냅니다.
- $d|n$ 는 d 가 n 의 약수라는 의미입니다.

2. $\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j)=1]$ (서로소 쌍의 개수)

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j)=1] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{d\mid\gcd(i,j)}\mu(d) \qquad \text{(by 1.)}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{d=1}^{n}\mu(d)[d\mid\gcd(i,j)]\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{d=1}^{n}\mu(d)[d\mid i][d\mid j]\\ &= \dots\end{aligned}$$

240

$$\dots = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{d=1}^{n}\mu(d)[d|i][d|j]$$

$$= \sum_{d=1}^{n}\left\{\mu(d)\left(\sum_{i=1}^{n}[d|i]\right)\left(\sum_{j=1}^{n}[d|j]\right)\right\}$$

$$= \sum_{d=1}^{n}\mu(d)\left\lfloor\frac{n}{d}\right\rfloor^{2}$$

3. $\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$ (모든 쌍의 gcd 합)

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}k[\gcd(i,j)=k] \\
&= \sum_{k=1}^{n}k\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j)=k] \\
&= \sum_{k=1}^{n}k\sum_{i'=1}^{\lfloor\frac{n}{k}\rfloor}\sum_{j'=1}^{\lfloor\frac{n}{k}\rfloor}[\gcd(i',j')=1] \qquad (i' := \frac{i}{k}, j' := \frac{j}{k}) \\
&= \dots
\end{aligned}
$$

$$\ldots = \sum_{k=1}^{n} k \sum_{d=1}^{\left\lfloor \frac{n}{k} \right\rfloor} \mu(d) \left\lfloor \frac{n}{kd} \right\rfloor^2 \qquad \text{(by 2.)}$$

$$= \sum_{l=1}^{n} \left\lfloor \frac{n}{l} \right\rfloor^2 \sum_{k|l} k\mu\left(\frac{l}{k}\right) \qquad (l := kd)$$

$$= \sum_{l=1}^{n} \left\lfloor \frac{n}{l} \right\rfloor^2 \phi(l)$$

$\phi(l)$ 함수는 오일러 피 함수$^{\text{Euler's totient function}}$를 의미합니다.

240

4. $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$

$$
\begin{aligned}
\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j) &= \sum_{n=1}^{N}\sum_{l=1}^{n}\left\lfloor\frac{n}{l}\right\rfloor^2\phi(l) \qquad \text{(by 3.)}\\
&= \sum_{l=1}^{N}\phi(l)\sum_{n=l}^{N}\left\lfloor\frac{n}{l}\right\rfloor^2\\
&= \sum_{l=1}^{N}\phi(l)\left\{l\sum_{s=1}^{\lfloor\frac{N}{l}\rfloor-1}s^2+\left\lfloor\frac{N}{l}\right\rfloor^2(1+N \bmod l)\right\}\\
&= \dots
\end{aligned}
$$

240

$$\dots = \sum_{l=1}^{N} \phi(l) \left\{ l \sum_{s=1}^{\left\lfloor \frac{N}{l} \right\rfloor - 1} s^2 + \left\lfloor \frac{N}{l} \right\rfloor^2 \left(1 + N - l \left\lfloor \frac{N}{l} \right\rfloor \right) \right\}$$

$$= \sum_{l=1}^{N} \phi(l) \hbar_2(N, l) \qquad (\hbar_2(N, l) := l \sum_{s=1}^{\left\lfloor \frac{N}{l} \right\rfloor - 1} s^2 + \left\lfloor \frac{N}{l} \right\rfloor^2 \left(1 + N - l \left\lfloor \frac{N}{l} \right\rfloor \right))$$

$\phi(l)$ 를 $\mathcal{O}(N)$ 전처리로 미리 구하고 매 루프마다 $\hbar_2(N, l)$ 를 $\mathcal{O}(1)$ 에 계산하면, $\mathcal{O}(N)$ 에 구할 수 있습니다.

$\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$ 의 값은 2단계에 걸쳐 구할 수 있습니다.

1. $\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$ (모든 쌍의 lcm 합)
2. $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$

240

1. $\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{lcm}(i,j)$ (모든 쌍의 lcm 합)

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{lcm}(i,j) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}[\gcd(i,j)=k]\frac{ij}{k} \qquad \left(\mathrm{lcm}(i,j)=\frac{ij}{\gcd(i,j)}\right)\\ &= \sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j)=k]\frac{ij}{k}\\ &= \sum_{k=1}^{n}\sum_{i'=1}^{\lfloor \frac{n}{k}\rfloor}\sum_{j'=1}^{\lfloor \frac{n}{k}\rfloor}[\gcd(i',j')=1]i'j'k \qquad \left(i':=\frac{i}{k}, j':=\frac{j}{k}\right)\\ &= \dots\end{aligned}$$

$$\dots = \sum_{k=1}^{n} \sum_{i'=1}^{\lfloor \frac{n}{k} \rfloor} \sum_{j'=1}^{\lfloor \frac{n}{k} \rfloor} \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} [d|i'][d|j']\mu(d)i'j'k \qquad \text{(by gcd-1.)}$$

$$= \sum_{k=1}^{n} k \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} \left\{ \mu(d) \left(\sum_{i'=1}^{\lfloor \frac{n}{k} \rfloor} [d|i']i' \right) \left(\sum_{j'=1}^{\lfloor \frac{n}{k} \rfloor} [d|j']j' \right) \right\}$$

$$= \sum_{k=1}^{n} k \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} \left\{ \mu(d) \left(\sum_{i''=1}^{\lfloor \frac{n}{kd} \rfloor} di'' \right) \left(\sum_{j''=1}^{\lfloor \frac{n}{kd} \rfloor} dj'' \right) \right\} \qquad (i'' := \frac{i'}{d}, j'' := \frac{j'}{d})$$

$$= \dots$$

240

$$\dots = \sum_{k=1}^{n} k \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} \left\{ d^2 \mu(d) \left(\sum_{i''=1}^{\lfloor \frac{n}{kd} \rfloor} i'' \right) \left(\sum_{j''=1}^{\lfloor \frac{n}{kd} \rfloor} j'' \right) \right\}$$

$$= \sum_{k=1}^{n} k \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} \left\{ d^2 \left(\frac{(\lfloor \frac{n}{kd} \rfloor)(\lfloor \frac{n}{kd} \rfloor + 1)}{2} \right)^2 \mu(d) \right\} \qquad \left(\sum_{i''=1}^{\lfloor \frac{n}{kd} \rfloor} i'' = \frac{(\lfloor \frac{n}{kd} \rfloor)(\lfloor \frac{n}{kd} \rfloor + 1)}{2} \right)$$

$$= \sum_{l=1}^{n} \left(\frac{(\lfloor \frac{n}{l} \rfloor)(\lfloor \frac{n}{l} \rfloor + 1)}{2} \right)^2 \sum_{d|l} \mu(d) l d \qquad (l := kd)$$

$$= \dots$$

240

$$\dots = \sum_{l=1}^{n} \left(\frac{(\lfloor \frac{n}{l} \rfloor)(\lfloor \frac{n}{l} \rfloor + 1)}{2} \right)^2 \nu(l) \qquad (\nu(l) := \sum_{d|l} \mu(d) l d)$$

$\nu(l)$ 는 multiplicative function 이고 $\nu(p^k) = p^k - p^{k+1}$ 임을 이용해 $\mathcal{O}(N)$ 전처리하면, 전체 식을 $\mathcal{O}(N)$ 에 계산할 수 있습니다.

2. $\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$

$$
\begin{aligned}
\sum_{n=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j) &= \sum_{n=1}^{N}\sum_{l=1}^{n}\left(\frac{(\lfloor\frac{n}{l}\rfloor)(\lfloor\frac{n}{l}\rfloor+1)}{2}\right)^2\nu(l) \\
&= \sum_{n=1}^{N}\sum_{l=1}^{n}\left(\left\lfloor\frac{n}{l}\right\rfloor^2+2\left\lfloor\frac{n}{l}\right\rfloor^3+\left\lfloor\frac{n}{l}\right\rfloor^4\right)\frac{\nu(l)}{4} \\
&= \sum_{l=1}^{N}\frac{\nu(l)}{4}\sum_{n=l}^{N}\left(\left\lfloor\frac{n}{l}\right\rfloor^2+2\left\lfloor\frac{n}{l}\right\rfloor^3+\left\lfloor\frac{n}{l}\right\rfloor^4\right) \\
&= \dots
\end{aligned}
$$

240

$$\ldots = \sum_{l=1}^{N} \frac{\nu(l)}{4} \left(\hbar_2(N,l) + 2\hbar_3(N,l) + \hbar_4(N,l)\right) \qquad \left(\hbar_a(N,l) := \sum_{n=l}^{N} \left\lfloor \frac{n}{l} \right\rfloor^a\right)$$

$\hbar_a(N,l)$는 gcd-4.와 비슷한 방식으로 전개하면, 매 루프마다 $O(1)$에 계산해서 전체 식을 $\mathcal{O}(N)$에 구할 수 있습니다.

수식전개가 **많이많이많이** 길지만, 수식전개를 하고 코딩으로 옮기는 것은 어렵지 않습니다.

```
ll s4 = ((((((((((1LL * (n / i - 1LL) * ....
```

다만, 오버플로우에 조심하세요! 중간중간 $\mathrm{mod}\ K$ 를 빠짐없이 해주어야합니다!

- 이 문제를 풀기 위해 단계적으로 아래의 문제를 풀면 좋습니다.

1. 폰친구 — BOJ 20296
2. Coprime Integers — BOJ 16409
3. 공약수 — BOJ 1792
4. LCM(i, j) — BOJ 11691
5. GCD vs LCM — BOJ 19138

240

F. 성싶당

ad_hoc, constructive
출제진 의도 – **Medium**

- 제출 66번, 정답 13팀 (정답률 19.697%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 44분
- 출제자 : 윤지학[cubelover], 박수현[shiftpsh]

문제를 조금 바꾸면 아래와 같이 쓸 수 있습니다. 서술의 편의를 위해 함수 d를 다음과 같이 정의합시다.

$$d(i,j) = \left(i\text{와 } j\text{를 각각 비트 문자열로 나타내었을 때 }\textbf{다른}\text{ 비트의 개수}\right)$$

문제는 다음 식을 **최대화**하는 길이 N의 서로 다른 비트 문자열 2^N 개의 permutation p를 구하는 것입니다.

$$\sum_{i=1}^{N-1} d(p_i, p_{i+1})$$

240

잠깐, 위 식을 최대화하는 대신 **최소화**할 수 있는지 먼저 생각해 볼까요?

- 서로 다른 비트 문자열의 permutation이므로, $i \neq j \Rightarrow d(i, j) \geq 1$입니다.
- 따라서 $d(p_i, p_{i+1})$의 값이 항상 1인 permutation이 존재한다면 그 때가 최소가 되는데$\cdots$

$\cdots$ 이런 permutation 중 하나로 **Gray code**가 있습니다. $N = 3$일 때의 Gray code는 아래와 같습니다.

$$000 \quad 001 \quad 011 \quad 010 \quad 110 \quad 111 \quad 101 \quad 100$$

Gray code는 1의 자리가 $0110\cdots$, 2의 자리가 $00111100\cdots$, 4의 자리가 $0000111111110000\cdots$ 로 반복되도록 하는 식으로 생성할 수 있습니다. 분할 정복 등의 방법을 사용 가능합니다.

또한, i번째 Gray code는 $i \oplus (i \gg 1)$이라는 사실을 이용할 수도 있습니다. $\oplus$는 비트 XOR, $\gg$는 오른쪽 시프트 연산입니다.

이제 $d\left(p_i, p_{i+1}\right)$의 합을 어떻게 최대화할 수 있는지 생각해 봅시다.

- 일단 일반성을 잃지 않고 $p_1 = 0$이라고 합시다.
- $i \neq j \Rightarrow d\left(i, j\right) \nleq N$이지만, $d\left(p_i, p_{i+1}\right)$의 값이 항상 N이 되도록 할 수는 없습니다. 이는 $i = \neg j$일 때에만 성립하기 때문입니다. $\neg$은 비트 NOT 연산입니다.
- $d\left(p_i, p_{i+1}\right)$의 값이 N 혹은 $N-1$만 등장하게 할 수 있을까요?

우선 길이 $N-1$의 비트 문자열으로 이루어진 Gray code를 생성하여 **홀수번째**에 배치합니다.

$$000 \quad 001 \quad 011 \quad 010$$

이후, 각각의 비트 문자열에 NOT 연산을 한 것을 **짝수번째**에 배치합니다.

$$000 \quad 111 \quad 001 \quad 110 \quad 011 \quad 100 \quad 010 \quad 101$$

240

$000 \quad 111 \quad 001 \quad 110 \quad 011 \quad 100 \quad 010 \quad 101$

- 홀수번째에 배치된 문자열들은 그 자체로 Gray code이므로, 0으로 시작하는 길이 N의 모든 비트 문자열을 포함합니다.
- 짝수번째에 배치된 문자열들은 홀수번째에 배치된 문자열들에 각각 NOT 연산을 취한 것이므로, 1로 시작하는 길이 N의 모든 비트 문자열을 포함합니다.
- 따라서 이는 2^N개의 서로 다른 비트 문자열의 permutation입니다.

$$000 \quad 111 \quad 001 \quad 110 \quad 011 \quad 100 \quad 010 \quad 101$$

- $d(p_{2i+1}, p_{2i+2}) = d(p_{2i+1}, \neg p_{2i+1}) = N$ 입니다.
- $d(p_{2i+2}, p_{2i+3}) = d(\neg p_{2i+1}, p_{2i+3}) = N - d(p_{2i+1}, p_{2i+3}) = N - 1$ 입니다. p_{2i+1} 의 다음 Gray code가 p_{2i+3} 이기 때문입니다.
- $d = N$ 인 쌍은 2^{N-1} 개로, 가능한 경우가 전부 등장했습니다. 이외의 모든 경우 $d = N - 1$ 이므로,

$$\sum_{i=1}^{N-1} d(p_i, p_{i+1}) = N2^{N-1} + (N-1)\left(2^{N-1} - 1\right)$$

이며, 이 값보다 큰 경우는 등장할 수 없습니다.

$d(a \oplus x, b \oplus x) = d(a, b)$ 이므로 위에서 구한 순열에 입력받은 최초항을 전부 비트 XOR해 주면, 문제의 조건을 모두 만족하는 순열을 만들 수 있습니다. 예를 들어 $p_1 = 101$ 이라면,

$$101 \quad 010 \quad 100 \quad 011 \quad 110 \quad 001 \quad 111 \quad 000$$

이 됩니다.

- 000 으로 시작하는 배열을 cyclic shift 해서 출력하는 방법도 있습니다. 다만 이 경우, $d\left(p_{2i+1}, p_{2i+2}\right) = N$ 이 되도록 **각별히 주의**해야 합니다. $d\left(p_{2i+2}, p_{2i+3}\right) = N$ 이 되도록 구현한 경우

$$
\begin{aligned}
\sum_{i=1}^{N-1} d\left(p_i, p_{i+1}\right) &= (N-1)\,2^{N-1} + N\left(2^{N-1} - 1\right) \\
&= (2N-1)\,2^{N-1} - N \\
&< (2N-1)\,2^{N-1} - N + 1 \\
&= N2^{N-1} + (N-1)\left(2^{N-1} - 1\right)
\end{aligned}
$$

으로, 최댓값이 아니게 됩니다.

240

G. 신촌지역 초중고등학생 프로그래밍 대회 동아리 연합 대회

`2_sat`

출제진 의도 – **Hard**

- 제출 9번, 정답 2팀 (정답률 22.222%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 127분
- 출제자 : 정연두`Green55`

240

✓ 문제 요약 : 이런 조건들을 만족하는 배열을 만들 수 있나요?

- $a_i = x$
- $a_i \,\&\, a_j = x$
- $a_i \mid a_j = x$
- $8 \le a_i \le 19$

- 더 쉬운 문제 부터 풀어봅시다.
- 이런 조건들을 만족하는 배열을 만들 수 있나요?
  - $a_i = x$
  - $a_i \,\&\, a_j = x$
  - $a_i \mid a_j = x$
  - $\mathbf{0} \le a_i \le \mathbf{1}$

- 이것은 대표적인 2-SAT 문제입니다.
- **2-SAT** : “$(a_i = 0/1) \vee (a_j = 0/1)$” 꼴의 식이 여러개 있을 때,
  이것들을 전부 만족하는 a를 찾아줍니다.
- 쉬운 문제의 조건들을, 위 형태의 식들을 이용해 동치가 되도록 나타내봅시다.
  - $a_i = x : (a_i = x) \vee (a_i = x)$
  - $a_i \,\&\, a_j = 0 : (a_i = 0) \vee (a_j = 0)$
  - $a_i \,\&\, a_j = 1 : (a_i = 1) \vee (a_i = 1), (a_j = 1) \vee (a_j = 1)$
  - $a_i \,|\, a_j = 0 : (a_i = 0) \vee (a_i = 0), (a_j = 0) \vee (a_j = 0)$
  - $a_i \,|\, a_j = 1 : (a_i = 1) \vee (a_j = 1)$

- 이제 우리는 $0/1$ 로는 문제를 쉽게 풀 수 있습니다.
- 따라서, 원래 문제도 비트로 쪼개서 생각해봅시다.
- $a_{i,j}$ 를 a_i 의 j 번째 (2^j) 비트라고 합시다.
- 예를 들어 $a_i = 18_{(10)} = 10010_{(2)}$ 이면 :

  $a_{i,0} = 0, a_{i,1} = 1, a_{i,2} = 0, a_{i,3} = 0, a_{i,4} = 1$

- 수의 범위가 $[01000_{(2)}, 10011_{(2)}]$ 이므로, 5 개의 비트만 고려해도 충분합니다.
- 예를 들어, $a_i \,\&\, a_j = x$ 는 다음과 동치입니다 : $a_{i,k} \,\&\, a_{j,k} = x_k (0 \le k \le 5)$
- 이런식으로 비트단위로 쪼개서 각 비트에 대해 “쉬운 문제” 를 풀어주면 됩니다.

- 하지만, 아직 우리는 $8 \le a_i \le 19$ 라는 조건은 만족하지 못했습니다.
- 이 조건도 2-SAT 형태의 식으로 나타낼 수 있을까요?

- $[8, 19]$ 의 정수를 이진수로 써보면 다음과 같습니다 :
  $01000_{(2)}, 01001_{(2)}, \ldots, 01110_{(2)}, 01111_{(2)}, 10000_{(2)}, 10001_{(2)}, 10010_{(2)}, 10011_{(2)}$

- 만약 4 번째 (2^4) 비트가 0 이면 :
  8 보다는 커야하기 때문에, 3 번째 (2^3) 비트는 반드시 1 이어야 합니다.
- 만약 4 번째 (2^4) 비트가 1 이면 :
  19 보다는 작아야하기 때문에, 2 번째 (2^2) 비트와 3 번째 (2^3) 비트는 반드시 0 이어야 합니다.

- 이런 관계들 역시 2-SAT의 식으로 나타낼 수 있습니다.
  - $a_{i,4}$가 0이면, $a_{i,3}$은 1이다 : $(a_{i,4} = 1) \lor (a_{i,3} = 0)$
  - $a_{i,4}$가 1이면, $a_{i,2}$은 0이다 : $(a_{i,4} = 0) \lor (a_{i,2} = 1)$
  - $a_{i,4}$가 1이면, $a_{i,3}$은 0이다 : $(a_{i,4} = 0) \lor (a_{i,3} = 1)$

- 드디어 모든 관계에 대한 식을 만들었습니다.
- $\mathcal{O}(N + M)$만에 해결 할 수 있습니다.

240

H. 카카오톡

`map, math`

출제진 의도 – **Easy**

- 제출 386번, 정답 30팀 (정답률 7.772%)
- 처음 푼 팀 : **app** (서강대학교), 13분
- 출제자 : 이국렬`lky7674`

- 여사건을 계산하는 것이 간단합니다.
- $\binom{n}{2} - ($같은 기울기 2개를 선택하는 경우의 수$)$.
- 같은 기울기의 직선의 개수는 a와 b를 $\gcd(a, b)$로 나누고 map에 저장해서 계산할 수 있습니다.

240

- 다만 이 문제는 코너 케이스가 많습니다.
- 대표적으로 $a < 0$ 또는 $b < 0$ 인 케이스가 있습니다. $(1, -1)$ 과 $(-1, 1)$ 을 같은 기울기로 계산해야 합니다.
- $a = 0$ 이거나 $b = 0$ 인 경우도 처리해야 합니다.

I. 팰린드롬 척화비

`constructive`

출제진 의도 – **Easy**

- 제출 60번, 정답 50팀 (정답률 85.000%)
- 처음 푼 팀 : **전생했더니 박건이었던 건에 대하여** (서강대학교), 2분
- 출제자 : 정기웅[QuqqU]

- 수미상관이면서 팰린드롬인, 아무 문자열을 출력하면 됩니다.
- 그런데 같은 문자로만 이루어진 문자열은, 수미상관이면서 팰린드롬입니다.
- 그래서 a를 N번 출력하면 쉽게 AC를 받을 수 있습니다.
- 참고로 99%의 팀에서 $aa \cdots a$로 AC를 받았습니다.

240

J. 의자 게임

`two_pointer`
출제진 의도 – **Hard**

- 제출 48번, 정답 4팀 (정답률 8.333%)
- 처음 푼 팀 : **No Orange cant win** (서강대학교), 115분
- 출제자 : 정연두[Green55]

240

- 게임의 상황이 복잡하므로, 하나씩 단순화 시켜봅시다.
- 매 초마다 오른쪽으로 회전하는 크기 N 의 배열 a 가 있습니다.

240

- 운영진은 “어느 순간”에, 원하는 원소를 a_N 의 뒤에 추가 할 수 있습니다.
- 하지만 어차피 배열은 회전하므로,
  참을성 있게 기다리면 원하는 위치에 원소를 추가 할 수 있습니다.
- 따라서, 운영진이 “아무 때나”
  원하는 곳에 원하는 원소를 추가 할 수 있다고 생각해도 무방합니다.

- 게임이 끝나려면 “어느 순간” 에 다음을 만족하는 연속 부분 수열이 있어야 합니다. :
  - $a_i, a_{i+1}, \ldots, a_{i+m-1}$ 을 정렬하면 $i, (i+1), \ldots, (i+m-1)$ 이 된다.
- “어느 순간” 이 아닌 “아무 때나” 다음을 만족하는 연속 부분 수열이 있으면 됩니다 :
  - 승리 조건 : a 를 원형으로 바꾼 배열 a' 에서 (즉, a'_N 과 a'_1 가 연속),
    $a'_i, a'_{i+1}, \ldots, a'_{i+m-1}$ 을 정렬하면 $k, (k+1), \ldots, (k+m-1)$ 이 된다.
    (k 는 임의의 정수, $a'_{N+1} = a'_1, a'_{N+2} = a'_2 \ldots$)
- a'_i 가 인덱스 k 에 올 때까지 기다리기만 하면 되니까요.

- 배열이 회전하는 것이 짜증나니, “아무 때나” 의 힘을 빌려 그냥 시간을 0초로 고정해버립시다.
- 즉, 배열 a' 는 더이상 회전하지 않고 초기 입력의 그대로입니다.

- 그렇다면 문제가 이렇게 단순화 되었습니다 :
  - 원형 배열 a' 가 주어진다. 원하는 곳에 원하는 원소를 최소로 추가하여, 승리 조건을 만족하는 연속 부분 수열이 존재하도록 만들어야한다.

- a' 에서 크기가 M 이하인 임의의 연속 부분 수열 $b = a'_i, ..., a'_j$ 를 고정했다고 생각 해봅시다.
- b에 $M - (j - i + 1)$ 개의 원소를 추가해 승리 조건을 만족할 조건은 다음과 같습니다 :
  - 승리 가능 조건 : b는 중복된 원소가 없고, $max(b) - min(b) < M$
- 운영진은 최소의 원소만 추가하여 승리 조건을 만족하고 싶으니 다음을 구하면 됩니다 :
  - a' 에서 승리 가능 조건을 만족하는 연속 부분 수열 b의 최대 길이는?

- 마지막으로, 짜증나는 원형 수열을 다시 일자로 펴봅시다.
- 원형이 아닌 일반 수열 $a'' = a_1, a_2, \ldots, a_n, a_1, a_2, ...a_n$ 로 문제를 풀어도, 원형 수열 a' 에서와 같은 답이 나옵니다.
- 어차피 $M \leq N$ 이고, 승리 가능 조건을 만족하려면 길이가 M 을 넘을 수 없기 때문에 a'' 에서 똑같은 원소를 여러번 사용할 일은 없기 때문입니다.

- 결국 우리가 풀 최종 문제는 다음과 같습니다 :
  - a'' 에서 승리 가능 조건을 만족하는 연속 부분 수열 b의 최대 길이는?

- 이제 드디어 좀 풀어볼만하게 문제가 간단해졌습니다!
- 연속 부분 수열은 $\mathcal{O}\left(N^2\right)$ 개나 있기 때문에 다 확인할 수는 없습니다.
- b가 승리 가능 조건을 만족하면, b의 연속 부분 수열도 승리 가능 조건을 만족합니다.

- 따라서 **투 포인터**를 이용 할 수 있으며,
  확인해야 할 연속 부분 수열의 후보가 $\mathcal{O}\left(N\right)$ 개로 줄어듭니다.
- 또는 확인할 b의 길이를 **이분탐색**하여 조건을 만족하는 최대값을 구하면,
  확인해야할 후보가 $\mathcal{O}\left(N \log N\right)$ 개로 줄어듭니다.

- 그렇다면 우리가 "확인 중인" 배열이 승리 가능 조건을 만족하는지 빠르게 확인하기 위해, 다음과 같은 연산을 $\mathcal{O}(1)$ 혹은 $\mathcal{O}(\log N)$ 정도에 지원하는 자료구조가 필요합니다.
  1. 원하는 원소 추가, 삭제
  2. 이미 특정 원소가 존재하는지 확인
  3. 현재 원소들의 최대값, 최소값은?
- 이것은 모노톤 큐, BBST, 세그먼트 트리 등의 자료구조를 사용하여 구현 가능합니다. C++의 std::set는 위의 연산이 전부 구현되어 있으므로 구현이 쉽습니다.

- 총 $\mathcal{O}(N), \mathcal{O}(N \log N), \mathcal{O}\left(N \log^2 N\right)$ 등의 시간에 구현 가능합니다.
- 시간 제한이 넉넉하기 때문에 C++ 기준 어느 복잡도여도 충분히 통과 가능합니다.

240

K. 합성인수분해

math, number_theory, primality_test

출제진 의도 – **Medium**

- 제출 318번, 정답 23팀 (정답률 7.233%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 25분
- 출제자 : 정연두Green55

240

- 먼저 N 을 소인수분해 해봅시다 :
  $N = P_1 \times P_2 \times \cdots \times P_M (P_i \leq P_{i+1}, P_i$ 는 소수$)$
- N 의 소인수는 최대 1개를 제외하고 $\sqrt{N}$ 이하임을 이용하면, $\mathcal{O}\left(\sqrt{N}\right)$ 만에 가능합니다.

- 이제 합성인수분해를 해봅시다.
- 수열이 사전순으로 가장 앞서려면, 앞에서부터 매번 최선을 다해 최대한 작게 만들어줍니다.
- 합성수가 되려면 최소 소수가 두개는 필요하니까, 앞에서부터 두개씩 묶어줍니다.
- $A = (P_1 \times P_2), (P_3 \times P_4), (P_5 \times P_6), \ldots$

- 이런식으로 두개씩 묶어주다가, 남은 P_i 가 딱 세개면 세개를 한번에 묶어야 합니다.
- 세개가 남았는데 두개만 묶으면, 소수가 딱 하나 남아 합성수가 되지 않기 때문입니다.

240

- 정리하면
  - $M = 1$ 이면, 합성인수분해 불가능
  - M 이 짝수면, $A = (P_1 \times P_2), (P_3 \times P_4), \ldots, (P_{M-3} \times P_{M-2}), (P_{M-1} \times P_M)$
  - M 이 홀수면, $A = (P_1 \times P_2), (P_3 \times P_4), \ldots, (P_{M-4} \times P_{M-3}), (P_{M-2} \times P_{M-1} \times P_M)$

- 가장 많이 틀리는 원인 : 6 의 합성인수분해는 6 입니다.

240

L. 습격받은 도시

ad_hoc, constructive, prefix_sum, sweeping

출제진 의도 – **Medium**

- 제출 95번, 정답 26팀 (정답률 28.421%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 12분
- 출제자 : 이상원gumgood

폭탄의 충격파는 건물 뿐만 아니라 건물 잔해에서도 멈추기 때문에 폭탄이 터지는 순서는 상관이 없습니다. 따라서 각 폭탄을 독립적으로 생각할 수 있습니다.

- ✓ 어떤 위치에 폭탄이 있었다면,
  해당 위치의 상하좌우 각 방향에 대해 가장 가까운 건물이 모두 건물 잔해가 되었어야 합니다.
- ✓ 각 위치에서 상하좌우 각 방향에 대해 가장 가까운 건물을 확인하여
  폭탄이 놓일 수 있는 위치인지 결정할 수 있습니다.
- ✓ 폭탄이 놓일 수 있는 위치를 적절히 골라 모든 건물 잔해에 충격파가 닿게 하면 됩니다.

폭탄의 위치를 적절히 고르는 여러가지 방법이 있을 수 있습니다. 그 중 가장 간단한 방법을 생각해봅시다.

- 폭탄의 충격파는 건물 잔해를 넘어갈 수 없습니다.
- 이로부터 건물 잔해에 서로 다른 폭탄의 충격파가 도착해도 된다는 사실을 알 수 있습니다.
- 따라서 폭탄을 놓을 수 있는 모든 위치에 놓아도 됩니다!

- 정리하면, 모든 위치에서 상하좌우에 대해 가장 가까운 건물이 모두 건물잔해가 되었는지 확인하여 폭탄을 놓으면 됩니다.
- Naive하게 $\mathcal{O}(N)$에 구현하면, 전체 시간복잡도는 $\mathcal{O}(N^3)$가 되어 시간초과를 피할 수 없습니다.
- **스위핑**, **누적합** 등으로 적절히 $\mathcal{O}(N^2)$에 구현하면 문제를 해결할 수 있습니다.

240

M. Go와 함께하는 전화망 서비스

`flow, mfmc`

출제진 의도 – **Challenging**

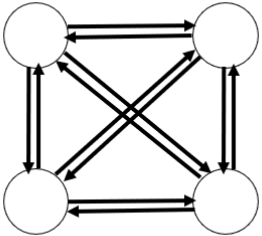
- 제출 9번, 정답 1팀 (정답률 11.111%)
- 처음 푼 팀 : **Supported by LKY** (연세대학교), 274분
- 출제자 : 이국렬 `lky7674`

- Degree Bounded Spanning Tree Polytope에 속하는지 확인하는 문제입니다.
- 해당 수식은 NP-Complete 문제인 Degree Bounded Minimum Spanning Tree의 근사해를 구하는데 사용됩니다.

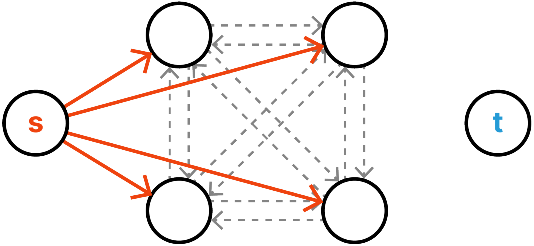
- 첫 번째와 세 번째 제약은 쉽게 확인할 수 있습니다.
- 두 번째 조건을 확인하는 게 상당히 까다롭습니다.

- 공집합이 아닌 모든 S에 대해서 $\sum_{e \in E(S)} x_e \leq |S| - 1$을 확인해야 합니다.
- $|S| - 1 - \sum_{e \in E(S)} x_e$의 최솟값이 0임을 확인하면 됩니다.
- 이를 조금 변형하면 $|V| + (|S| - 1) - \sum_{e \in E(S)} x_e$의 최솟값이 $|V|$임을 확인하면 됩니다.

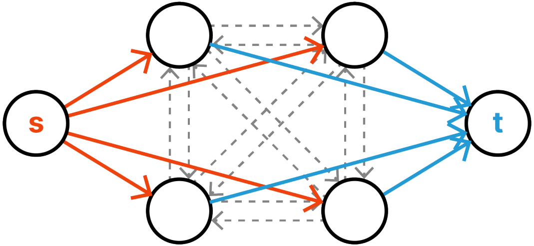
- 저희는 $|V| - 1 = \sum_{e \in E} x_e$ 임을 알고 있습니다.
- 때문에 해당 식을 $|S| + \sum_{e \in \delta(S)} x_e + \sum_{e \in E(V-S)} x_e$ 로 변환할 수 있습니다.
- 이후 해당 수식을 $|S| + \frac{1}{2} \sum_{e \in \delta(S)} x_e + \frac{1}{2} \sum_{v \notin S} \sum_{e \in \delta(v)} x_e$ 로 정리하도록 하겠습니다.
- 이렇게 하면 min cut을 통해서 식의 최솟값을 구할 수 있습니다.



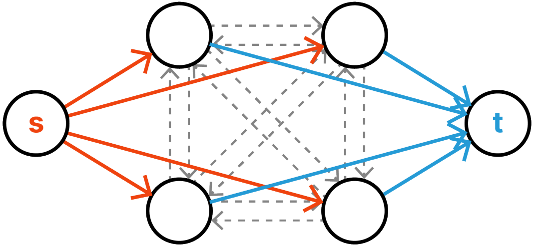
우선 기존 완전 그래프의 간선의 capacity를 $\frac{1}{2}x_e$로 설정합니다.



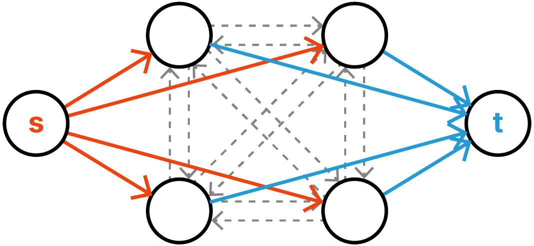
Source와 각 정점으로 이동하는 간선의 capacity를 $\frac{1}{2}\sum_{e\in\delta(v)} x_e$로 설정합니다.



각 정점에서 sink로 이동하는 간선의 capacity를 1로 설정합니다.



여기서 s-t cut, $(\{s\} \cup S, \{t\} \cup (V - S))$ 의 cut capacity를 계산하면
$|S| + \frac{1}{2} \sum_{e \in \delta(S)} x_e + \frac{1}{2} \sum_{v \notin S} \sum_{e \in \delta(v)} x_e$ 가 됨을 알 수 있습니다.



놓치기 쉬운 부분이 있는데, 이 그래프에서 min-cut을 구하게 되면 틀리는데, 이는 $|V| \geq 1$임을 간과했기 때문입니다. 그래서 s에서 각각의 정점 v로 가는 간선의 capacity를 무한대로 설정한 것들의 min cut을 구하셔야 합니다.

✓ 비슷하지만 좀 더 쉬운 문제들로 연습해봅시다.

- 물건 가져가기 5 BOJ 19579
- 영웅은 죽지 않아요 4 BOJ 12936
- Maximum Profit 4 BOJ 17345