# STPC 2024 Autumn by Seoultech FLY

## Editorial

Seoultech FLY

| | Problem | Difficulty | Author |
|---|---|---|---|
| A | 바코드 닉네임 | **Bronze** | 노현제hjroh0315 |
| B | Max-Queen | **Bronze** | 노현제hjroh0315 |
| C | MEX vs OR | **Silver** | 노현제hjroh0315 |
| D | $x$와 $x+1$의 차이 | **Silver** | 노현제hjroh0315 |
| E | ABB to BA (Easy) | **Silver** | 노현제hjroh0315 |
| F | ABB to BA (Hard) | **Gold** | 노현제hjroh0315 |
| G | 수열과 개구리 | **Gold** | 노현제hjroh0315 |
| H | 잘못 생성된 데이터 | **Platinum** | 노현제hjroh0315 |
| I | 올바른 괄호 문자열과 쿼리 | **Diamond** | 노현제hjroh0315 |
| J | 서울과 학기-술 대학교 | **Diamond** | 노현제hjroh0315 |

# A. 바코드 닉네임

`#implementation`

출제진 의도 — **Bronze**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `menborong`, +1분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부]sjh1224`, +0분

## A. 바코드 닉네임

- 문제에서 요구하는 대로 구현하면 됩니다.
- 현재 읽고 있는 문자가 `l`이라면 `L`을, `I`라면 `i`를 출력하면 해결됩니다.
- 또는, 입력 제한을 무시하고 Python의 `swapcase()`를 사용해도 좋습니다.

# B. Max-Queen

`#arithmetic`, `#greedy`

출제진 의도 — **Bronze**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `tlsgusrb314`, +4분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] Sengoku Nadeko`, +2분

## **B.** Max-Queen

- 한 행에 있는 퀸의 수가 증가한다면, 공격하는 쌍의 수는 반드시 증가합니다.
- 이는 한 행의 퀸의 수가 $k$ $(k > 1)$이라면 공격하는 쌍의 수가 $k - 1$임에 따라 증명됩니다.
- 따라서, 한 행에는 가능한 한 많은 퀸을 두는 것이 최선입니다.
- 이 논리는 한 열, 한 대각선에도 마찬가지로 적용됩니다.
- 따라서, 모든 칸에 퀸을 두는 것이 최선입니다.

- 모든 칸에 퀸을 두었을 때의 공격하는 쌍의 수는 다음과 같이 구해집니다.

$$4nm - 3(n + m) + 2$$

- 이는 각 칸이 기여하는 공격하는 쌍의 수를 구하여 계산할 수 있습니다.
- $n = 1$ 또는 $m = 1$이 입력에 존재한다면 예외처리가 필요합니다.
- 하지만, 입력 제한에 따라 $2 \leq n, m$이 보장되므로 예외처리는 필요하지 않았습니다.

# C. MEX vs OR

`#bitmask`, `#implementation`, `#ad_hoc`

출제진 의도 — **Silver**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `tlsgusrb314`, +6분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] Xakep`, +3분

## C. MEX vs OR

- ✓ 구현에 따라 지문에 주어진 대로 구현해도 문제를 해결할 수 있습니다.
- ✓ 정렬을 하여 $\mathrm{mex}$를 구할 경우 $\mathcal{O}(r \log r)$, 정렬하지 않을 경우 $\mathcal{O}(r^2)$에 구해집니다.
- ✓ 다음 쪽부터 $\mathcal{O}(1)$ 해법을 서술합니다.

## C. MEX vs OR

- 비트 OR은 다음과 같은 성질을 가집니다.
  - 두 정수 중 한 쪽이 양의 정수라면, 비트 OR의 결과도 양의 정수입니다.
  - 두 정수 중 한 쪽이 $0$이라면, 비트 OR의 결과는 다른 한 쪽과 같습니다.
- 함수 $\mathrm{mex}$는 다음과 같은 성질을 가집니다.
  - 입력으로 양의 정수만이 주어진다면, $0$이 포함되지 않으므로 항상 결과는 $0$입니다.
- 이 두 성질을 활용할 수 있습니다.

- $l > 0$인 경우를 가정해 봅시다.
  - 입력은 이미 모두 양의 정수입니다.
  - 입력으로 양의 정수만이 주어진 함숫값은 $0$이 되어야만 합니다.
  - 따라서 정답은 $0$입니다.
- $x > 0$인 경우를 가정해 봅시다.
  - 모든 항에서 적어도 한 쪽인 $x$가 양의 정수였으므로, 모든 항은 양의 정수입니다.
  - 따라서 입력은 모두 양의 정수입니다.
  - 입력으로 양의 정수만이 주어진 함숫값은 $0$이 되어야만 합니다.
  - 따라서 정답은 $0$입니다.

- 이제 남은 경우는 $l = x = 0$인 경우입니다.
  - 이 경우 $0$부터 $r$까지의 정수가 모두 입력으로 주어집니다.
  - 입력으로 주어지지 않은 가장 작은 음이 아닌 정수는 $r + 1$입니다.
  - 따라서 정답은 $r + 1$입니다.
- 세 가지 경우 모두 각각 $\mathcal{O}(1)$ 시간에 해결해 주었으므로, 시간 복잡도는 $\mathcal{O}(1)$입니다.

# D. $x$와 $x+1$의 차이

`#math`, `#number_theory`

출제진 의도 — **Silver**

- 출제자: 노현제[hjroh0315]
- 온라인 오픈 콘테스트
  - 최초 해결: `chmpro`, +3분
- 오프라인 온사이트 대회
  - 최초 해결: [외부] Xakep, +5분

## D. $x$와 $x+1$의 차이

- 예제를 잘 살펴봅시다. 무언가 보이나요?
- 다음의 가설을 고려하고, 증명함을 통해 문제를 해결할 수 있습니다.
  - $\left\lfloor \frac{x}{k} \right\rfloor$와 $\left\lfloor \frac{x+1}{k} \right\rfloor$의 값이 다르게 되는 $k$는 반드시 $x+1$의 약수입니다.
  - 또한, 그 역도 성립합니다.

## D. $x$와 $x+1$의 차이

- 이 가설을 약수의 관점이 아니라 배수의 관점에서 바라봅시다.
- $k$가 고정되고 $a$가 바뀔 때, $\left\lfloor \frac{a}{k} \right\rfloor$가 바뀌는 시점은 $a$가 $k$의 배수일 때입니다.
- 따라서, $a$가 $k$의 배수가 아니었으나 $a+1$은 배수일 때에만 $\left\lfloor \frac{a}{k} \right\rfloor$와 $\left\lfloor \frac{a+1}{k} \right\rfloor$가 다릅니다.
- 이제 이 관찰을 가지고 약수의 관점으로 돌아와서 봅시다.

## D. $x$와 $x+1$의 차이

- $x$가 $k$의 배수가 아니었으나 $x+1$이 $k$의 배수일 때 두 값은 다릅니다.
- $k \neq 1$이면 두 수가 동시에 $k$의 배수가 될 수는 없습니다. 두 수는 서로소이기 때문입니다.
- $k = 1$이면 $x \neq x+1$에 따라 반드시 두 값이 다릅니다.
- $x+1$이 $k$의 배수라는 것은 $k$가 $x+1$의 약수라는 의미입니다.
- 따라서 가설이 증명되었습니다.
- $x+1$ 자신을 제외한 $x+1$의 모든 약수를 오름차순으로 출력하면 문제가 해결됩니다.

# E. ABB to BA (Easy)

`#implementation`, `#string`

출제진 의도 — **Silver**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `serenity_tranquility`, +16분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] ychangseok`, +8분

## **E.** ABB to BA (Easy)

- 다음 사실을 관찰합시다.
  - ABB가 BA로 바뀔 때, 문자열의 길이는 $1$만큼 감소합니다.
  - 따라서, ABB가 BA로 바뀌는 사건은 $n$번을 초과하여 발생할 수 없습니다.

## **E.** ABB to BA (Easy)

- 문제의 제한상 $\mathcal{O}(n^2)$ 시간 복잡도로 해결할 수 있습니다.
- 문자열에서 ABB 하나를 찾아 BA로 바꾸는데 $\mathcal{O}(n)$ 시간이 걸립니다.
- 이 사건이 $\mathcal{O}(n)$번 발생하므로 단순하게 해결하면 $\mathcal{O}(n^2)$ 시간이 걸리게 됩니다.
- **Python의 `replace` 함수는 이 문제에 쓸 수 있는 함수가 아님에 유의해야 합니다!**

# F. ABB to BA (Hard)

`#ad_hoc`, `#data_structures`, `#linked_list`

출제진 의도 — **Gold**

- 출제자: 노현제[hjroh0315]
- 온라인 오픈 콘테스트
  - 최초 해결: `tlsgusrb314`, +16분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] Xakep`, +8분

## **F.** ABB to BA (Hard)

- 모든 동작이 마무리된 후의 문자열을 처음부터 생성하려고 시도해 봅시다.
- 문자열 $T$에 $S$에 포함된 글자를 하나씩 순서대로 추가합니다.
- 만약 문제가 “ABB를 모두 삭제할 것"을 요구했다면, 스택으로 해결할 수 있습니다.
- 그러나 새로운 BA가 새로운 ABB를 만들 수 있어 스택으로 직접 해결하기 어렵습니다.

## F. ABB to BA (Hard)

- 만약 문자열 $T$가 ...AB꼴이었는데, B가 추가되어 ABB가 만들어졌다고 합시다.
- 그렇다면 $T$는 ...BA꼴로 바뀝니다. 이때 여기서 다음의 관찰을 합시다.
  - 우리는 항상 "맨 앞의" ABB를 BA로 바꾸었습니다.
  - 따라서 접두사 ...에는 A, AB는 포함될 수 있으나 ABB는 있을 수 없습니다.
  - 만약 접두사에 ABB가 있었다면 그 이전에 BA로 바뀌었어야 했기 때문입니다.
- 그러므로, 새로운 BA로 인해 생길 수 있는 최초의 ABB는 새로운 BA 바로 앞입니다.
- 그 ABB가 다시 변환된 BA가 새로운 ABB를 만들 수 있지만, 위의 사실은 바뀌지 않습니다.

## **F.** ABB to BA (Hard)

- ✓ 이제 ABB가 BA로 바뀐 직후, 첫 번째 새로운 ABB가 존재할 수 있는 위치는 명확합니다.
- ✓ 이 위치에 ABB가 있는지 찾고 BA로 바꾸는 것은 연결 리스트로 $\mathcal{O}(1)$에 가능합니다.
- ✓ 검색해야 할 위치는 마지막 위치와 인접하므로 포인터가 이동하는 시간은 $\mathcal{O}(1)$입니다.
- ✓ 시간 복잡도 분석을 위해 분할상환분석을 사용해야 합니다.

## **F.** ABB to BA (Hard)

- ✓ $T$에 문자를 추가하는 사건이 $\mathcal{O}(n)$번 발생합니다.
- ✓ E번의 관찰에 의해 ABB가 BA로 바뀌는 사건은 $\mathcal{O}(n)$번 발생합니다.
- ✓ ABB를 찾으려고 하였으나 ABB가 없어 탐색을 종료한 사건의 수는 $\mathcal{O}(n)$번 발생합니다.
- ✓ 각 사건은 한 번에 $\mathcal{O}(1)$ 시간이 소요됩니다.
- ✓ 따라서 총 시간 복잡도는 $\mathcal{O}(n)$입니다.

# G. 수열과 개구리

`#shortest_path`, `#dijkstra`

출제진 의도 — **Gold**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: adhd, +19분
- 오프라인 온사이트 대회
  - 최초 해결: [외부] Xakep, +12분

- 두 수열을 그래프 형태로 구성하면 다음과 같습니다.
  - 양쪽의 수열 밖 영역을 정점 $0, n+1$으로 만듭시다.
  - 각 정점 $i$ $(1 \le i \le n)$에는 $\max(0, i - a_i)$로 향하는 가중치 $b_i$의 간선이 있습니다.
  - 각 정점 $i$ $(1 \le i \le n)$에는 $\min(n+1, i + a_i)$로 향하는 가중치 $b_i$의 간선이 있습니다.
- 이 그래프의 모든 정점에서 정점 $0$ 또는 $n+1$로 가는 최단 거리를 찾아야 합니다.

## G. 수열과 개구리

- ✓ 모든 정점에서 종점으로 가는 최단 거리를 구하기 위해 다음 테크닉을 사용합니다.
  - ✓ 모든 간선을 뒤집은 뒤, 종점을 시점으로 취급하여 최단 거리를 구합니다.
- ✓ 이 테크닉은 **BOJ 1238: 파티** 등에서 사용된 바 있습니다.

## G. 수열과 개구리

- 이 문제에서 고려해야 할 종점은 $0$과 $n+1$로 $2$개입니다.
- 두 정점에서 각각 다익스트라 알고리즘을 수행해도 됩니다.
- 두 정점을 모두 우선순위 큐에 넣어서 다익스트라 알고리즘을 시작해도 됩니다.
- 시간 복잡도는 $\mathcal{O}(n \log n)$입니다.

# H. 잘못 생성된 데이터

`#bayes`, `#probability`, `#implementation`, `#data_structures`

출제진 의도 — **Platinum**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `menborong`, +57분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] Xakep`, +25분

## H. 잘못 생성된 데이터

- 어떤 순열이 주어지든, 알고리즘 A가 이를 생성할 확률은 $\frac{1}{1000!}$으로 동일합니다.
- 알고리즘 B에 대해 시뮬레이션을 시행해 알고리즘 B가 생성할 확률을 구합시다.

## H. 잘못 생성된 데이터

- 베이즈 정리 $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$를 잠시 생각합시다.
- 이에 따라 $P(\textbf{알고리즘} \mid \textbf{순열}) = \frac{P(\text{순열} \mid \text{알고리즘})P(\text{알고리즘})}{P(\text{순열})}$입니다.
- 주어진 순열을 알 때 $P(\textbf{순열})$과 $P(\textbf{알고리즘})$은 이 문제에서 상수입니다.
- 따라서 $P(\textbf{알고리즘} \mid \textbf{순열})$이 높다는 것은 $P(\textbf{순열} \mid \textbf{알고리즘})$이 높다는 것입니다.

## H. 잘못 생성된 데이터

- ✓ 따라서 알고리즘 B가 주어진 순열을 생성할 확률이 $\frac{1}{1000!}$보다 높은지 판단해야 합니다.
- ✓ 시뮬레이션에서 적절한 자료구조를 통해 각 원소가 매 차례 이동한 방향을 찾습니다.
- ✓ 이동한 방향에 따라서 곱해질 확률을 의사코드로 인해 알고 있습니다.
- ✓ 따라서, 시뮬레이션을 통해 확률을 구하는 것도 충분히 가능합니다.

## H. 잘못 생성된 데이터

- ✓ 슬프게도, `double`이나 `long double`의 정밀도는 $\frac{1}{1000!}$를 표현하기에 너무 낮습니다.
- ✓ Python의 `Decimal` 등 임의 정밀도 실수를 사용하기에는 시간이 충분하지 않습니다.
- ✓ 그러므로, 정밀도를 약간 희생하더라도 대소관계를 비교할 방법을 찾아야 합니다.

## H. 잘못 생성된 데이터

- 확률 $p$ 자체 대신 $\log(p)$를 구하는 것을 시도해봅시다.
- $\log\left(\frac{1}{n!}\right) = -\log(n!)$은 그 절댓값이 $\mathcal{O}(n \log n)$이므로, `double`에 저장이 가능합니다.
- $p$를 계산할 때 곱셈만을 사용했으므로, $\log(p)$를 계산할 때는 덧셈만을 사용합니다.
- 이렇게 하면, 정밀도를 희생하였지만 **맞았습니다!!**를 받기에는 충분하게 됩니다.
- 시간 복잡도는 $\mathcal{O}(n \log n)$ 또는 $\mathcal{O}(n \log^2 n)$입니다.

## H. 잘못 생성된 데이터

- ✓ 그런데, 이렇게 끝낼 문제라기엔 해설에 무언가 더 있어야 할 것 같습니다.
- ✓ 이 해설에서 정확도를 증명하지 않았다는 점을 고려하면 더욱 그렇습니다.

✓ **다음 쪽부터 진짜 정해를 소개합니다.**

# H+. 잘못 생성된 데이터 (裏)

`#bayes`, `#ad_hoc`, `#dp`, `#precomputation`

출제진 의도 — **Platinum**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `menborong`, +57분
- 오프라인 온사이트 대회
  - 최초 해결: `[외부] Xakep`, +25분

## $H^+$. 잘못 생성된 데이터 (裏)

- ✓ 확률을 직접 계산할 필요가 있을까요?
- ✓ 확률 자체보다 간단하면서 알고리즘을 유의미하게 대변하는 확률변수를 찾아 봅시다.

- Inversion은 어떨까요?
  - Inversion의 개수는 의미가 있을 것 같지만, 문제가 있습니다.
  - Inversion의 개수는 알고리즘 A에서 정규분포를 따릅니다.
  - 서로 다른 두 정규분포에서 뽑은 두 확률변수는 서로 구별하기 쉽지 않습니다.
  - 따라서 Inversion의 개수는 쓰기에 좋은 확률변수는 아닙니다.
- 이러한 면에서 쓰기에 간단하면서 유의미한 확률변수를 찾아보겠습니다.

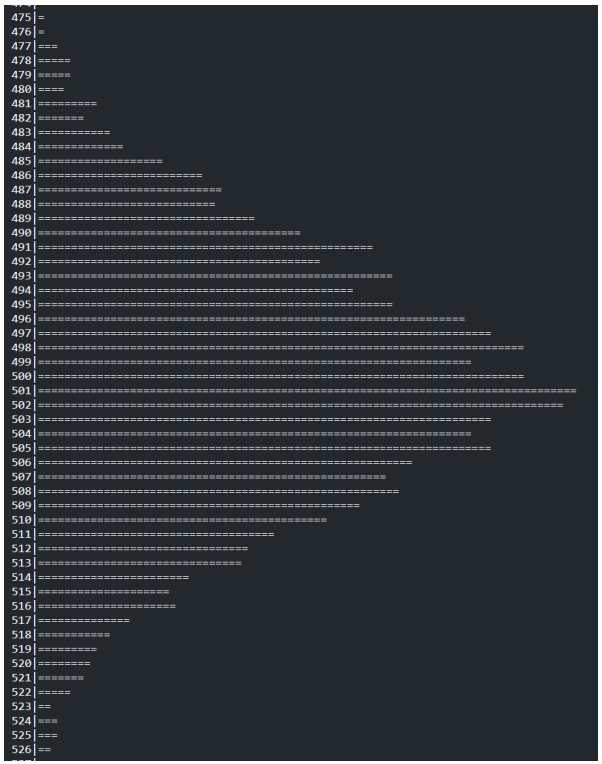
## H+. 잘못 생성된 데이터 (裏)

- 놀라울 정도로 단순하면서도 큰 의미를 가지는 확률변수가 있습니다.
- 바로 **1의 위치**입니다.

# H+. 잘못 생성된 데이터 (裏)

- ✓ 알고리즘 A에서, 각 위치에 $1$이 등장할 확률은 당연히 $\frac{1}{1000}$으로 균일합니다.
- ✓ 알고리즘 B는 어떨까요? 직접 $5000$개 정도의 표본을 뽑아 도수분포표를 그려 봅시다.

# H+. 잘못 생성된 데이터 (裏)


475|=
476|=
477|===
478|=====
479|=====
480|====
481|=========
482|=======
483|===========
484|============
485|==================
486|========================
487|=========================
488|========================
489|================================
490|======================================
491|================================================
492|=======================================
493|================================================
494|=============================================
495|================================================
496|============================================================
497|================================================================
498|====================================================================
499|================================================================
500|====================================================================
501|========================================================================
502|======================================================================
503|==================================================================
504|================================================================
505|==================================================================
506|================================================
507|=============================================
508|================================================
509|=============================================
510|=========================================
511|================================
512|==============================
513|============================
514|======================
515|==================
516|====================
517|=============
518|==========
519|========
520|========
521|=======
522|=====
523|==
524|===
525|===
526|==

- ?????????????

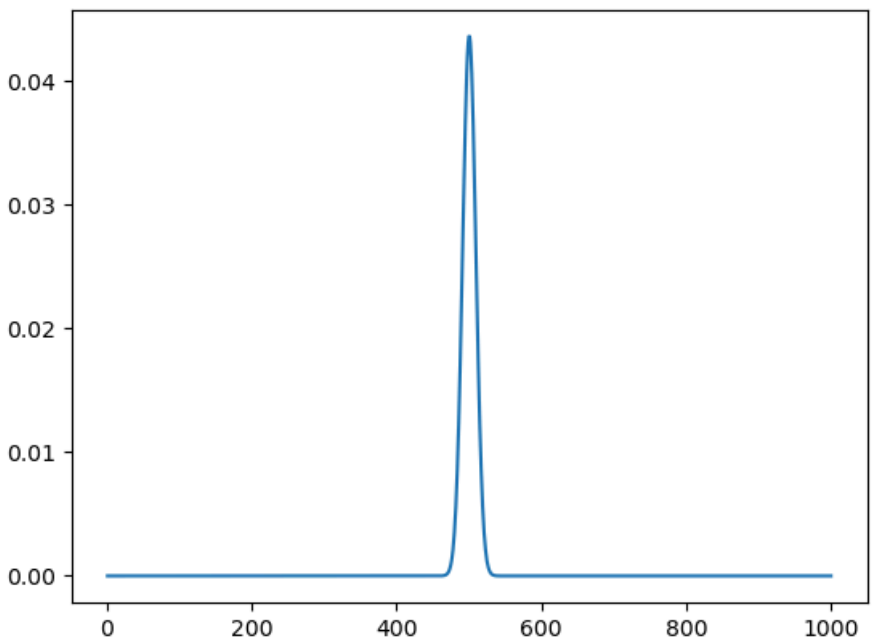
## H+. 잘못 생성된 데이터 (裏)

- 분명히 알고리즘 A는 모든 위치에서의 확률이 $\frac{1}{1000}$이었습니다.
- 그런데 알고리즘 B는 도수가 $470$ 이상 $530$ 이하에 집약되어 있습니다!
- 또한, $i < 400$이나 $i > 600$의 구간에서는 도수가 보이지도 않습니다!
- 이제 **1의 위치**를 좋은 확률변수로 활용할 수 있다는 확신이 있습니다.

- ✓ 어떤 위치에서 $1$이 등장할 확률은 DP로 미리 구할 수 있습니다.
- ✓ $\mathrm{dp}_{n,i} = \mathrm{dp}_{n-1,i} \cdot q_{n-1,i} + \mathrm{dp}_{n-1,i-1} \cdot p_{n-1,i-1}$ 꼴의 점화식을 만들 수 있습니다.
- ✓ DP 테이블은 $\mathcal{O}(n^2)$개의 셀에 각각 $\mathcal{O}(1)$개의 전이가 있습니다.
- ✓ 따라서 DP 테이블을 $\mathcal{O}(n^2)$ 시간에 완전히 채울 수 있습니다.

## H+. 잘못 생성된 데이터 (裏)

- DP 테이블로 얻은 확률을 그래프로 그리면 다음과 같습니다.




- 확률이 $\frac{1}{1000}$보다 큰 구간으로 $i \in [476, 525]$를 얻습니다.

- ✓ 이렇게 얻은 Python 솔루션은 다음과 같습니다.

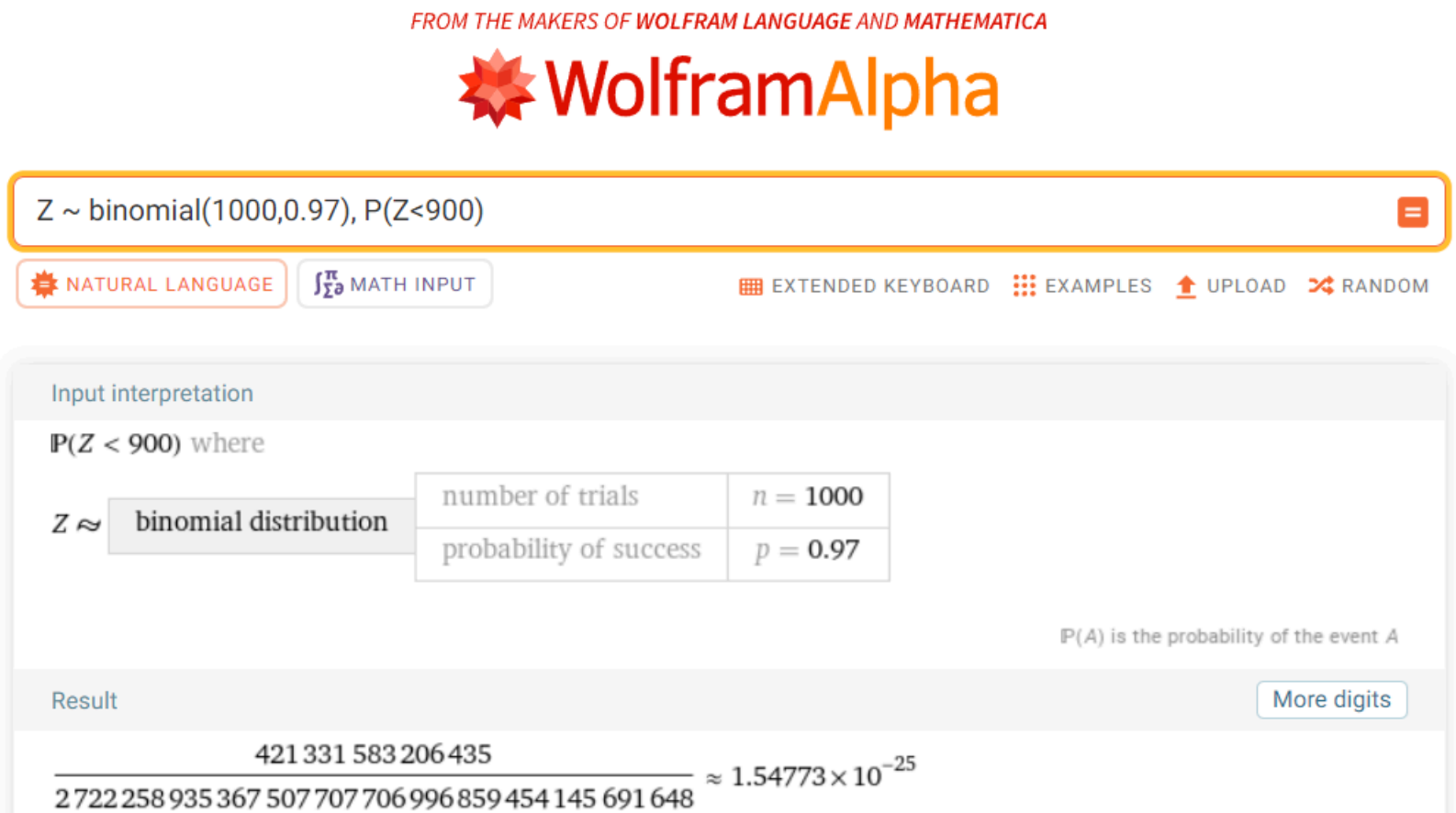
```
for i in range(int(input())):
  n=int(input())
  print(["Alice","Bob"][475<=[*map(int,input().split())].index(1)<525])
```

- ✓ 이 소스 그대로 **맞았습니다!!**를 받을 수 있고, 시간 복잡도는 $\mathcal{O}(n)$입니다.

## H+. 잘못 생성된 데이터 (裏)

- DP테이블에서 얻은 확률로 실패 확률도 계산할 수 있습니다.
  - 확률이 $\frac{1}{1000}$보다 높은 도수에 대해서는 (B로 판정했으므로) $\frac{1}{2} \cdot \frac{1}{1000}$을 더해줍니다.
  - 확률이 $\frac{1}{1000}$보다 낮은 도수에 대해서는 (A로 판정했으므로) $\frac{1}{2} \cdot p$를 더해줍니다.
- 그 결과 하나의 순열에 대한 실패 확률이 약 $2.808\%$임을 구할 수 있습니다.

# H⁺. 잘못 생성된 데이터 (裏)


FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA
WolframAlpha
Z ~ binomial(1000,0.97), P(Z<900)
NATURAL LANGUAGE
MATH INPUT
EXTENDED KEYBOARD
EXAMPLES
UPLOAD
RANDOM
Input interpretation
ℙ(Z < 900) where
Z ≈ binomial distribution
number of trials
n = 1000
probability of success
p = 0.97
ℙ(A) is the probability of the event A
Result
More digits
421 331 583 206 435 / 2 722 258 935 367 507 707 706 996 859 454 145 691 648 ≈ 1.54773 × 10^−25

## H+. 잘못 생성된 데이터 (裏)

- ✓ 따라서 입력 $150$개 중 적어도 하나라도 틀릴 확률은 $1-\left(1-1.6\cdot 10^{-25}\right)^{150}$ 미만입니다.
- ✓ 이 값은 약 $2.4\cdot 10^{-23}$에 해당합니다.
- ✓ 따라서 이 솔루션이 틀릴 확률은 충분히 낮습니다.

# I. 올바른 괄호 문자열과 쿼리

`#data_structures`, `#segtree`

출제진 의도 — **Diamond**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: `menborong`, +82분
- 오프라인 온사이트 대회
  - 최초 해결: [외부] Xakep, +36분

- ✓ 출제자가 영감을 받은 문제는 **LeetCode 678. Valid Parenthesis String**입니다.
- ✓ 링크된 문제에는 쿼리가 없으나, 이 문제에는 쿼리가 있습니다.
- ✓ 링크된 문제의 $\mathcal{O}(n)$ 해법을 알면 이 문제를 푸는데 도움을 받을 수 있을까요?

- 링크된 문제에는 다음과 같은 $\mathcal{O}(n)$ 해법이 있습니다.
  - 스택에 남은 괄호의 수를 세어 판정하는 알고리즘을 변형합니다.
  - 스택에 남은 괄호의 수로 가능한 값은 항상 하나의 구간 $[L, R]$을 이룹니다.
  - (가 들어올 경우 구간은 $[L, R]$에서 $[L+1, R+1]$로 바뀝니다.
  - )가 들어올 경우 구간은 $[L, R]$에서 $[\max(L-1, 0), R-1]$로 바뀝니다.
  - $*$가 들어올 경우 구간은 $[L, R]$에서 $[\max(L-1, 0), R+1]$로 바뀝니다.
  - 이때 매 순간 $0 \leq R$이 성립해야 하며, 종료 후 $L = 0$이 성립해야 합니다.
- 이 솔루션을 세그먼트 트리에 접목시킬 수 있을까요?

- ✓ $R$의 경우에는 $1$과 $-1$로 이루어진 배열의 누적 합으로 생각하면 됩니다.
- ✓ 이는 **금광 세그먼트 트리**로 익히 알려진 세그먼트 트리의 응용으로 해결됩니다.
- ✓ 그런데 $L$에 대해서는 어떻게 해결해야 할까요?

- ✓ **ReLU 세그먼트 트리**를 소개합니다.

- ReLU 세그먼트 트리는 다음의 부분 문제를 해결하기 위해 사용할 수 있습니다.
  - 수열 $a$와 $b$가 있습니다.
  - 쿼리 `change` $i$ $x$ $y$는 $a_i = x, b_i = y$로 값을 변경하는 쿼리입니다.
  - 쿼리 `query` $l$ $r$ $x$는 $(f_r \circ f_{r-1} \circ \cdots \circ f_{l+1} \circ f_l)(x)$를 구하는 쿼리입니다.
    - 이때 $f_i$는 $f_i(x) = \max(x - a_i, b_i)$로 정의됩니다.

- 세그먼트 트리의 각 노드가 일차함수를 저장하는 경우에 대해 생각합시다.
- 두 일차함수가 주어질 때, 그 합성 또한 일차함수입니다.
- 또한, 함수의 합성은 결합법칙을 만족합니다.
- 이제 일차함수의 합성은 모노이드를 이루므로, 세그먼트 트리에서 사용할 수 있습니다.
- ReLU 세그먼트 트리는 어떨까요?

- ✓ 함수 $f(x) = \max(x - a, b)$와 $g(x) = \max(x - c, d)$를 합성한 결과를 관찰합시다.
- ✓ $x \leq c + d$일 때, 함숫값은 상수입니다.
- ✓ $x - c \leq a + b$일 때, 함숫값은 상수입니다.
- ✓ $x > a + b + c$이고 $x > c + d$일 때, 기울기는 $1$이 됩니다.
- ✓ 따라서 함수 $(f \circ g)(x)$는 기존의 두 함수와 같은 꼴을 가집니다!

- ✓ 함수를 합성한 결과와 그 기울기에 따라, $(f \circ g)$의 식을 추론할 수 있습니다.
- ✓ 함수가 꺾이는 $x$좌표는 $X = \max(a + b + c, c + d)$가 됩니다.
- ✓ $x$가 $-\infty$로 발산할 때, $y$좌표는 $Y = \max(b, d - a)$가 됩니다.
- ✓ 이에 따라 합성함수의 식이 다음과 같음을 추론할 수 있습니다.

$$(f \circ g)(x) = \max(x - X + Y, Y)$$

- ✓ $\max(x-a, b)$꼴의 함수의 합성이 모노이드임을 증명했습니다.
- ✓ 이제 함수를 노드에 저장하는 세그먼트 트리를 사용해 부분 문제를 해결할 수 있습니다.

- $L$에 대한 부분 문제의 해결을 다음과 같이 처리할 수 있습니다.
  - $L \geq 0$이 항상 성립하므로, $L + 1 = \max(L + 1, 1)$입니다.
  - 이제 ReLU 세그먼트 트리를 사용할 수 있습니다.
  - 트리의 각 리프 노드에는 $\max(L + 1, 1)$ 또는 $\max(L - 1, 0)$이 있습니다.
  - 쿼리를 통해 구간의 함수들을 합성한 합성함수를 찾을 수 있습니다.
  - 이 함수에 $0$을 대입해보면 됩니다.
- 이제 $L$과 $R$에 대한 부분 문제를 모두 해결했으므로, 전체 문제를 해결할 수 있습니다.

# J. 서울과 학기-술 대학교

`#dp`, `#divide_and_conquer_optimization`

출제진 의도 — **Diamond**

- 출제자: 노현제hjroh0315
- 온라인 오픈 콘테스트
  - 최초 해결: -, +?분
- 오프라인 온사이트 대회
  - 최초 해결: -, +?분

## J. 서울과 학기-술 대학교

- ✓ 수강할 과목을 잘 선택해 분자와 분모의 비율을 최대화해야 하는 문제입니다.
- ✓ 과목의 개수만 제한되어 있다면 **이분 탐색으로 해결할 수 있음이 알려져 있습니다.**
- ✓ 그러나 이 문제에서 제한은 과목의 개수가 아니라 총 학점입니다.

- ✓ 총 학점 $h_1 + h_2 + \cdots + h_k$의 값을 고정합시다.
- ✓ 그렇다면 최대화해야 할 것은 분자 $h_1p_1 + h_2p_2 + \cdots + h_kp_k$뿐입니다.
- ✓ $\mathrm{dp}[w] = ($**총 가중치가 $w$일 때 최대 가치**$)$를 사용하는 DP 문제에는 무엇이 있을까요?

- ✓ 그렇습니다. 이 문제는 냅색 문제였습니다!
- ✓ 그러나 일반적인 냅색 문제는 해결하는데 $\mathcal{O}(NW)$ 시간이 걸립니다.
- ✓ 이 문제를 해결하기 위해 중요한 제한 하나에서 힌트를 얻어야 합니다.

- ✓ 모든 과목에 대해 $1 \leq H \leq 15$가 성립합니다. 이는 어떤 의미일까요?
- ✓ $H$의 값 하나에 대해 $\mathcal{O}(L)$ 시간만이 걸리도록 하는 방법이 있을까요?

- 수열 $A$, $B$가 주어질 때 다음 조건을 만족하는 수열 $C$를 구하는 방법을 생각합시다.

$$C_i = \max_{0 \le j \le i} \left( A_j + B_{i-j} \right)$$

- 이 부분 문제를 **Max-Plus Convolution**이라고 부릅니다.

- $A$ 또는 $B$가 위로 볼록할 경우 이를 빠르게 해결하는 방법이 알려져 있습니다.
- 예를 들어, Li-Chao Tree로 $\mathcal{O}(N \log N)$에 해결하거나...
- 분할 정복을 사용한 최적화로 $\mathcal{O}(N \log N)$에 해결하거나...
- SMAWK algorithm으로 $\mathcal{O}(N)$에 해결할 수 있습니다.

- 하나의 무게에 대한 냅색 문제를 이상의 부분 문제로 쪼갤 수 있습니다.
- $B_i = (i$**개를 가져갈 때 최대 가치**$)$로 정의합니다.
- 이 $B_i$의 값은 정렬 후 그리디하게 구하면 됩니다.
- $B$는 단조감소하는 수열의 누적합이므로, $B$는 위로 볼록합니다.

- ✓ $\mathrm{dp}_i$를 $i \bmod w$가 같은 인덱스끼리 모아 $A_i$로 정합니다.
- ✓ 이제 $C_i = \max_j\left(A_j + B_{i-j}\right)$를 구하면 해당 인덱스에서의 새로운 값이 됩니다.
- ✓ 이를 모든 $i \bmod w$의 값에 대해 수행합니다.
- ✓ 시간 복잡도는 $\mathcal{O}\left(\frac{W}{w} \cdot w\right) = \mathcal{O}(W)$가 됩니다.
- ✓ 이를 $k$가지 다른 무게에 대해 시행하면, $\mathcal{O}(n \log n + kW)$로 냅색이 해결됩니다!

- ✓ 이제 냅색 문제의 부분을 $\mathcal{O}(n \log n + HL)$에 해결했습니다.
- ✓ 나머지는 단순히 RMQ를 해결할 수 있는 자료구조로 해결하면 됩니다.
- ✓ 정해 코드의 총 시간 복잡도는 $\mathcal{O}(n \log n + HL + L \log L + q)$입니다.