2018 아주대학교 프로그래밍 경시대회

풀이 슬라이드

1A/2A. 맞았는데 왜 틀리죠?

출제자: 김현정

- sample_fail = 샘플 테스트케이스 S1개 중 하나라도 틀렸다면 True
- system_fail = 시스템 테스트케이스 S2개 중 하나라도 틀렸다면 True

APC 2018

1A/2A. 맞았는데 왜 틀리죠?

출제자: 김현정

- sample_fail = 샘플 테스트케이스 S1개 중 하나라도 틀렸다면 True
- system_fail = 시스템 테스트케이스 S2개 중 하나라도 틀렸다면 True

- if(!sample_fail && !system_fail) => Accepted
- else if(sample_fail) => Wrong Answer
- else => Why Wrong!!!

APC 2018

1A/2A. 맞았는데 왜 틀리죠?

출제자: 김현정

- sample_fail = 샘플 테스트케이스 S1개 중 하나라도 틀렸다면 True
- system_fail = 시스템 테스트케이스 S2개 중 하나라도 틀렸다면 True

- if(!sample_fail && !system_fail) => Accepted
- else if(sample_fail) => Wrong Answer
- else => Why Wrong!!!

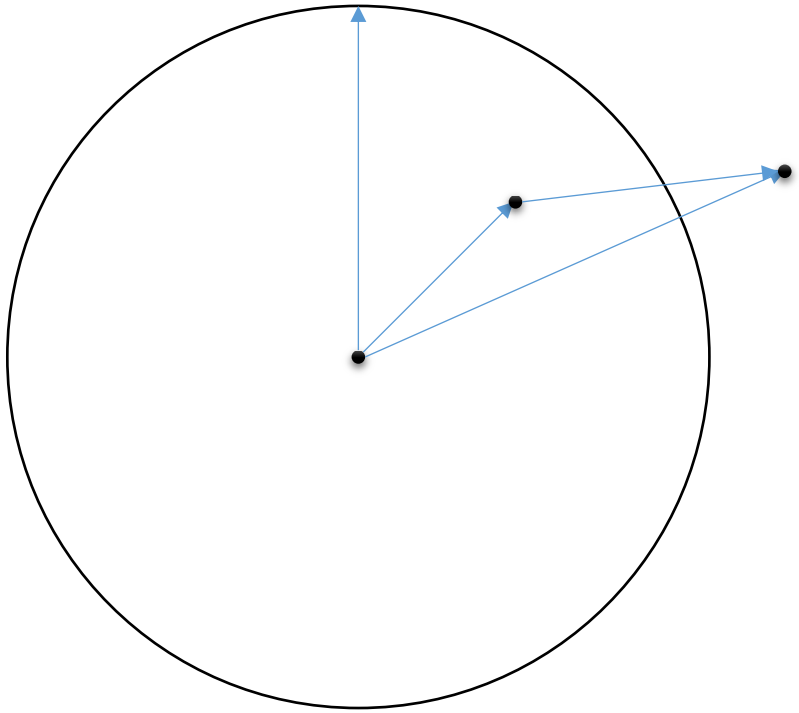
- 시간복잡도 O(N)
- 이제 이 문제를 풀었으니 맞왜틀은 충분한 검증 후에 하도록 합시다.

APC 2018

1B/2E. 낚이고 낚아라

출제자: 김동이

- 다각형의 모든 영역이 원의 내부에 들어오는지 판단하기 위해서는?
  - 원의 중심부터 가장 먼 거리의 점 만을 보면 된다.



- 원 외부로 벗어나는 영역이 존재하기 위해서는 최소 하나의 점은 원 밖에 있어야 한다.
- 중심부터 가장 먼 점이 원안에 있다면?
  ⇔ 어떤 점도 원 밖에 없다 ⇔ 해당 도형은 원 안에 포함된다.
- 즉, 어떤 도형을 원안에 포함하기 위한 '최소의 반지름'은 가장 먼 점까지의 거리다.

APC 2018

1B/2E. 낚이고 낚아라

출제자: 김동이

- 그러므로 최소 K개의 도형을 포함하고 싶다면?
  - '가장 먼 점까지의 거리'가 R이하가 되는 도형이 K개 이상이 되도록 R을 정하면 된다.
  - ⇔ k번째로 필요 낚시거리가 작은 도형을 포함할 수 있으면 된다.
- 각 도형별로 필요한 '낚시 거리'를 계산한다.
- 도형별 낚시거리를 오름차순으로 정렬하여 k번째 거리를 낚시거리 R로 설정한다.

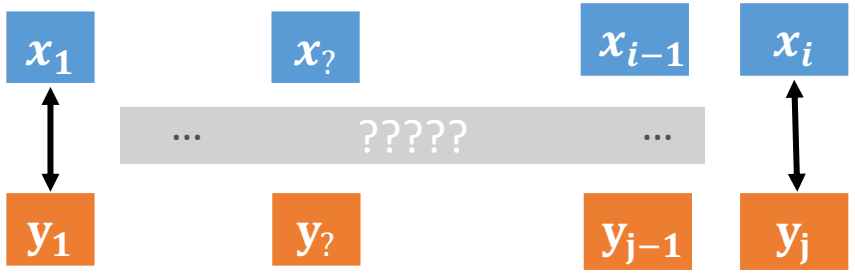
APC 2018

1C. Ah-Choo!

출제자: 김동이

- DTW(Dynamic Time Wrapping) 알고리즘에 대한 설명은 모두 문제에 주어져 있다!
- 재귀적인 점화식을 설계할 수 있다.
- 대응 관계는 교차되지 않으므로, 앞에 있는 시점끼리 대응된 이후에는 미래의 시점과 대응되지 않는다.

- F(i,j)를 다음과 같이 정의해보자
  - ⇔ X(i), Y(j)를 대응시켰을 때, X(1) ~ X(i)와 Y(1) ~ Y(j)모두 대응시키는 최소 거리



APC 2018

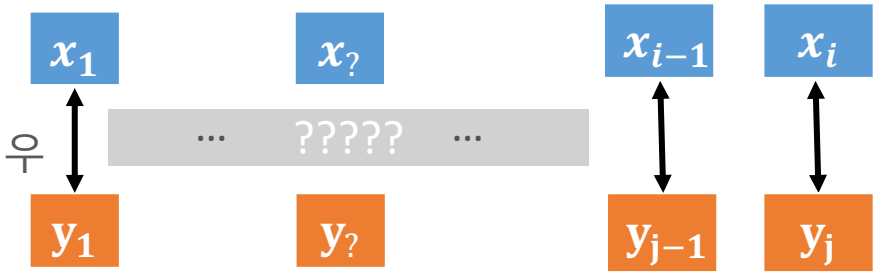
1C. Ah-Choo!

출제자: 김동이

- Case 1

$X(i)$와 $Y(j)$가 모두 이전의 $X(i-1)$과 $Y(j-1)$과 전혀 대응되지 않는 경우

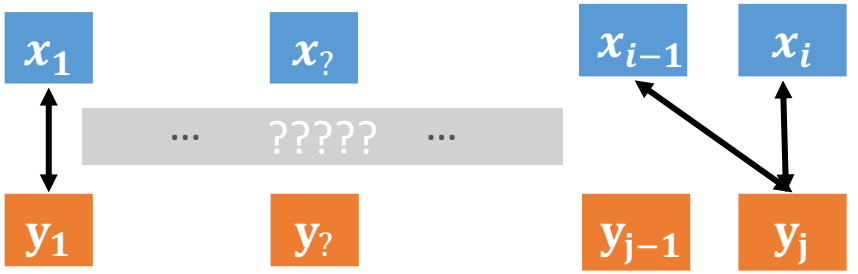
$\Leftrightarrow F(i,j) = F(i-1,j-1) + \{X(i) - Y(j)\}^2$



- Case 2

$X(i-1)$과 $Y(j)$가 대응되는 경우

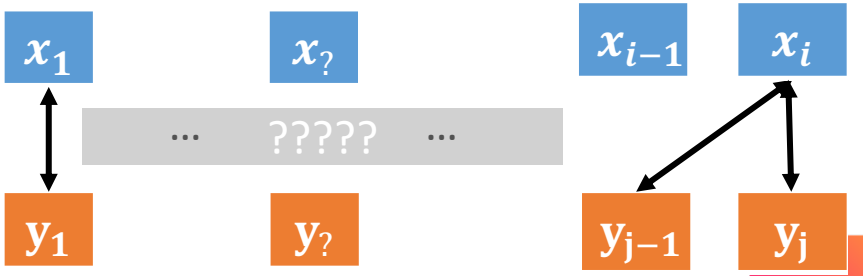
$\Leftrightarrow F(i,j) = F(i-1,j) + \{X(i) - Y(j)\}^2$



- Case 3

$X(i)$과 $Y(j-1)$가 대응되는 경우

$\Leftrightarrow F(i,j) = F(i,j-1) + \{X(i) - Y(j)\}^2$



2018

1C. Ah-Choo!

출제자: 김동이

- 결과적으로 아래와 같은 재귀식을 설계할 수 있다.

$$F(i,j) = \{X(i) - Y(j)\}^2 + \begin{cases} F(i-1,j-1) \\ F(i-1,j) \\ F(i,j-1) \end{cases}$$

- 모든 파라미터 i, j가 각각 1~N사이의 자연수이므로 모든 파라미터 조합에 대한 상태공간을 정의할 수 있다.
- 즉 Memoization을 사용해 Dynamic Programming을 설계할 수 있다.

APC 2018

1D. 카드 팩 구매하기

출제자: 김동이

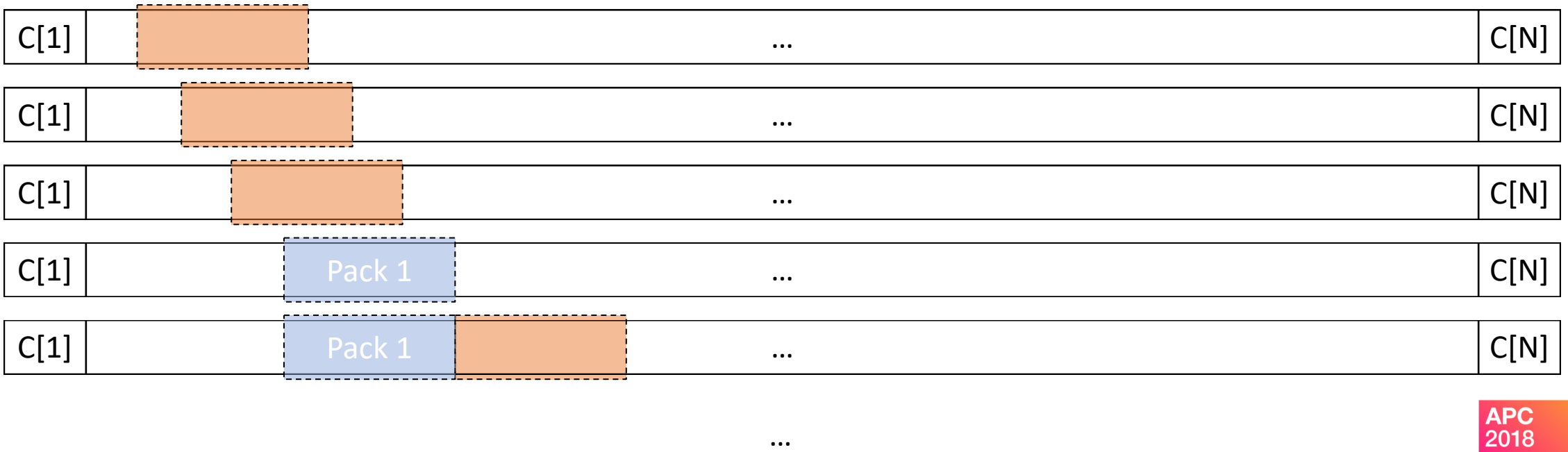
- 카드 팩의 크기(각 카드 팩에 포함된 카드 수)를 변수로 둘 경우, 각 카드 팩의 경계나 크기를 결정하기 복잡해진다.
- But, 카드 팩의 크기가 정해져 있다면?
  - 해당 크기로 카드 팩 들을 구매할 수 있는지 없는지 판단하는 것은 쉽다.

APC 2018

1D. 카드 팩 구매하기

출제자: 김동이

- 카드 팩의 크기가 정해 졌을 때, 규칙을 만족하는 가장 많은 카드 팩을 만드는 방법은?
  - 왼쪽부터 연속한 k개의 범위를 검사해 나간다.
  - 서로 다른 k개의 카드를 가진 범위를 찾는다면, 해당 범위는 하나의 카드 팩으로 설정하고 건너뛴다.
- 사이즈 k인 슬라이딩 윈도우 기법으로 만들 수 있는 카드 팩의 수를 계산한다.



APC 2018

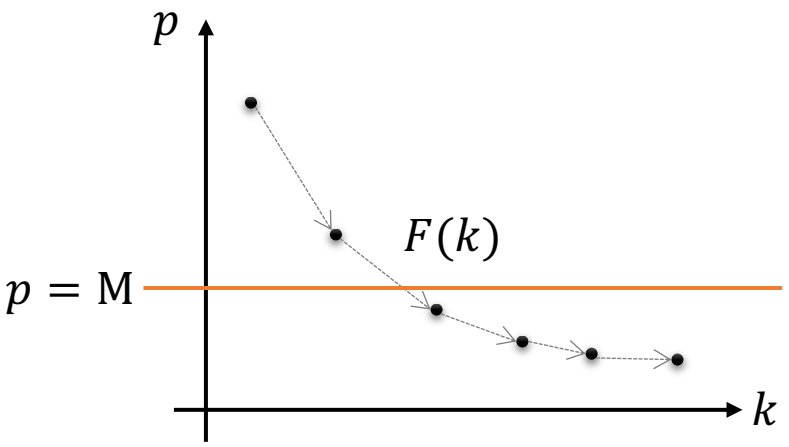
1D. 카드 팩 구매하기

출제자: 김동이

- 카드 팩의 크기가 커질 수록, 최대로 만들 수 있는 카드 팩의 수는 작아진다.

$p = F(k)$ :: 카드 팩의 크기가 k일 때 만들 수 있는 최대 카드 팩의 수

- 위와 같은 함수를 정의하면 단소 감소함수가 됨을 알 수 있다.



$F(k) \geq M$ 을 만족하는 최소의 k를 찾으면 된다.

∴ 변수 k에 대한 Binary Search로 정답을 찾을 수 있다.

모범답안 : https://gist.github.com/waps12b/9edcad62c5265208512b0d6a50fba1bc

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Small:
- 주헌이가 먹을 메뉴 조합의 최소값과 최대값을 지정한다.

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Small:
- 주헌이가 먹을 메뉴 조합의 최소값과 최대값을 지정한다.

- i번째로 스코빌지수가 작은 메뉴를 최소값으로 가지고
- j번째로 스코빌지수가 작은 메뉴를 최대값으로 가지는 메뉴의 수?
- => i, j번째 메뉴는 꼭 포함하고 (i+1 ~ j-1)메뉴로 구성되는
  모든 부분집합 개수: 총 2^{j-i-1} 가지

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Small:
- 2^0~2^N의 값을 미리 구해두고, 스코빌지수를 소팅한다.
- 이중반복문을 통해 최소값 i와 최대값 j를 정해
- 모든 (i, j) 쌍에 대해 (val[j] – val[i]) * 2^{j-i-1}의 합이 정답

- 시간복잡도: $O(N^2)$

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Large:
- 모든 조합의 (최대값 – 최소값)의 합은
- (모든 조합의 최대값의 합) – (모든 조합의 최소값의 합)과 같다.

- 최소값과 최대값을 함께 지정하지 말고, 따로 구하자.

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Large:
- i번째 스코빌지수를 최대값으로 가지는 모든 메뉴 조합의 개수:
- => 2^{i-1}가지
- i번째 스코빌지수를 최소값으로 가지는 모든 메뉴 조합의 개수:
- => 2^{N-i}가지

APC 2018

1E/2G. 너 봄에는 캡사이신이 맛있단다

출제자: 김현정

- Large:
- 각 i번째 스코빌지수에 대해 val[i] * ($2^{i-1} - 2^{N-i}$)의 합이 정답

- 음수가 나올 수 있는 변수에 대한 모듈러는 아래와 같이 처리합시다.
- => ans = (ans + add_value % MOD + MOD) % MOD

- 시간복잡도: O(NlogN)
  - sort: O(NlogN)
  - compute answer: O(N)

APC 2018

1F. System Call

출제자: 이주명

- 버퍼의 크기가 증가함에 따라 read 함수의 호출 횟수도 변할 것이다. 이렇게 변할 수 있는 경우는 O(√F)이다.
  - 파일의 크기를 F이라고 하면 [F/1], [F/2], ..., [F/[√F]], [F/[√F]]-1, [F/[√F]]-2, ..., 1이다. 여기서 [.]는 올림 함수이다. [F/[√F]]≤[√F]이기 때문에 전체 경우의 수는 2√F 이하가 된다.
    - F/[√F]≤√F이므로 [F/[√F]]≤[√F]도 성립한다.

APC 2018

1F. System Call

출제자: 이주명

- 각 파일에 대해 read 함수의 호출 횟수가 변하는 구간마다 몇 개씩 감소하는지를 따로 구할 수 있다.
  - 버퍼의 크기가 $[\sqrt{F}]$보다 작거나 같을 때는 직접 계산이 가능하다.
  - read 함수의 호출 횟수가 n+1에서 n이 되는 버퍼의 크기 중 최솟값을 x이라고 하자. 그러면 $[F/(x-1)]=n+1$이고 $[F/x]=n$이다. 앞의 식으로부터 $F/(x-1)>n$을 추론할 수 있고, 뒤의 식으로부터 $F/x\le n$을 추론할 수 있다. 이를 다시 쓰면 $x-1<F/n$, $x\ge F/n$이다. $x-1<F/n\le x$을 만족하는 정수 x는 $[F/n]$이다.

APC 2018

1F. System Call

출제자: 이주명

- 마지막에 가능한 모든 버퍼의 크기에 대해 read 함수의 호출 횟수를 알 수 있다. 이 값을 바탕으로 프로그램의 수행 시간을 계산할 수 있고, 그 중 가장 작은 경우가 정답이다.
  - 수행 시간 = 함수의 호출 횟수 x (버퍼의 크기 + T)

APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표

Small 부터 보자

- N이 1이라면 ?
  - 현재 위치 1에서 목적지 D까지의 최소 cost를 구해야 한다.
  - dp[cur] : cur 까지의 min cost 라 정의
  - 이를 저장할 배열 dp[] 를 선언한다.
  - dp[1] = 0 , 외의 모든 값을 INF 으로 초기화

Sample 1 : N = 1, D = 12, L = 3, x[]={8, 2, 5} 예시

0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
1	2	3	4	5	6	7	8	9	10	11	12

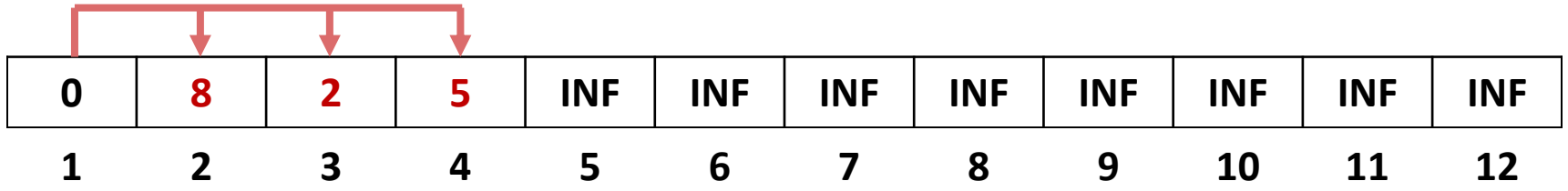
APC 2018

1G. 남제 어드벤쳐

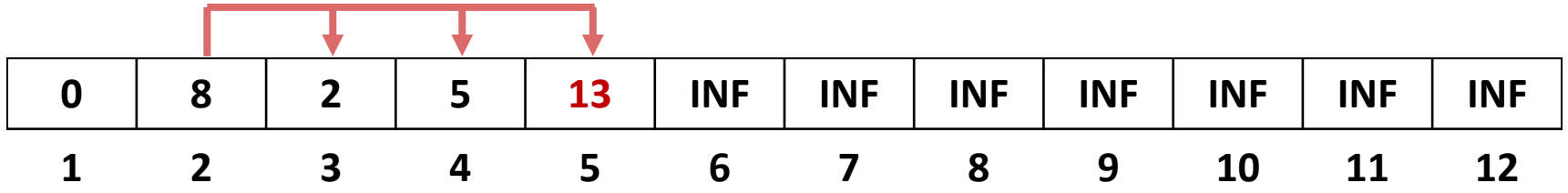
출제자: 홍준표

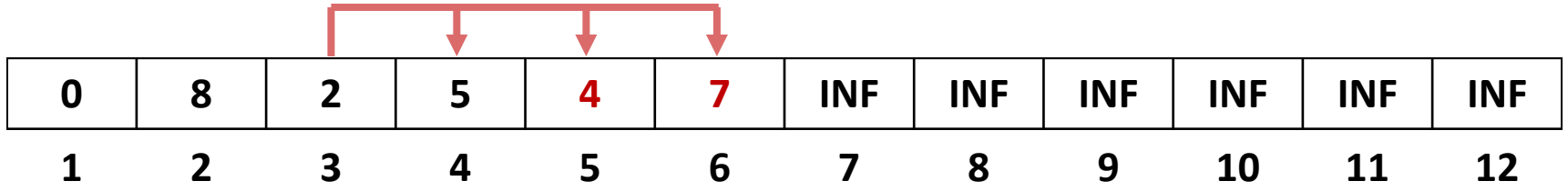
현재 위치 1에서 갈 수 있는 다음의 L칸을 min cost로 갱신, 현재위치는 다음으로 넘어간다.

도착한 다음칸은 다시 갱신될 일이 없으므로 최적임을 보장한다.



dp[cur+i] = min(dp[cur+i],dp[cur]+x[i]) 의 점화식을 반복하면 ..





APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표

최종적으로 다음과 같은 테이블을 얻을 수 있다.

코딩의 편리함을 위해 dp[] 배열을 L개 더 여유롭게 주면 좋다.

0	8	2	5	4	7	6	9	8	11	10	13	12	15
1	2	3	4	5	6	7	8	9	10	11	12		

dp[D] : D 까지의 min cost 이므로 답은 dp[D]!

시간복잡도 : O(DL)

APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표

N이 1 이상 이라면 ?

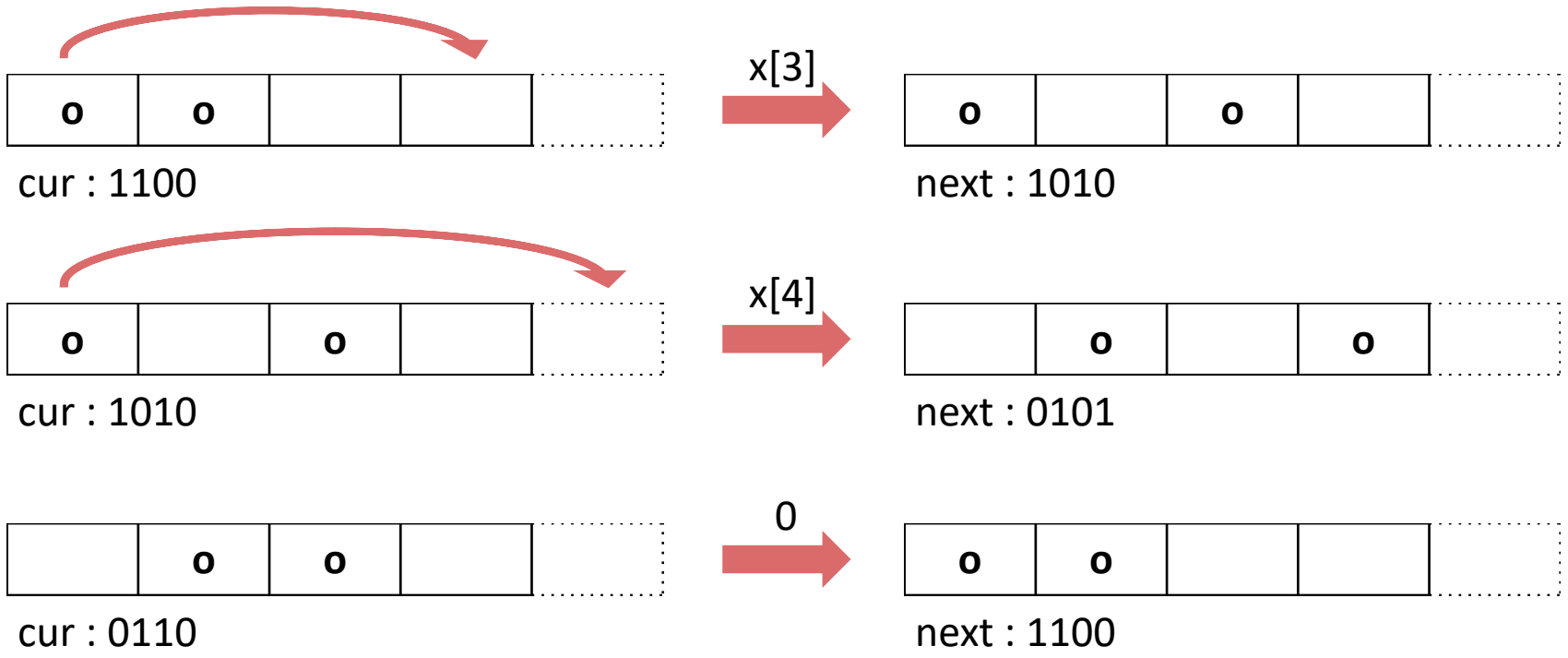
- L개의 칸에 사람이 있음을 bit를 통해 표현할 수 있다.
- 첫 번째 사람이 이동한 상태 next 또한 bit를 통해 표현할 수 있다.
- 이렇게 발생하는 코스트를 mat[cur][next] 에 표현할 수 있다.
- 만들어진 mat : 1 만큼 이동했을 때의 상태 -> 상태 별 최소 코스트 라 정의할 수 있다.

APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표

N이 1 이상 이라면 ?



- 첫 번째 칸에 옮겨갈 사람이 없으므로 shift만 한다. 이 때의 cost는 0.

APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표

1 칸 만큼 이동했을 때의 최소 코스트 mat 을 구했다.

- 우리의 목적은 D 만큼 이동했을 때의 최소 코스트 mat
- 구해둔 mat 에 mat 을 곱하면 어떨까 ?

플로이드-워셜 알고리즘

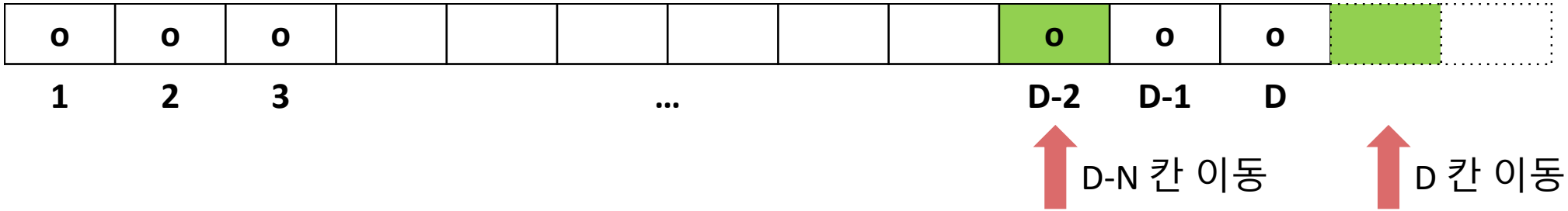
mat^2[cur][next] = min(mat[cur][mid] + mat[mid][next])

- 상태는 $2^{12} = 4096$ 에 3승을 더..?
- 내가 봐야할 상태는 ${}_{L}C_{N}$ 개 뿐 ! ${}_{12}C_{3} = 220$

APC 2018

1G. 남제 어드벤쳐

출제자: 홍준표



목표는 mat^{D-N}[cur][cur]

- 큰 수 거듭 제곱 아이디어로 log(D-N) 번만 곱셈 수행
- 시간복잡도 : $O\left(\binom{L}{N} logD\right)$

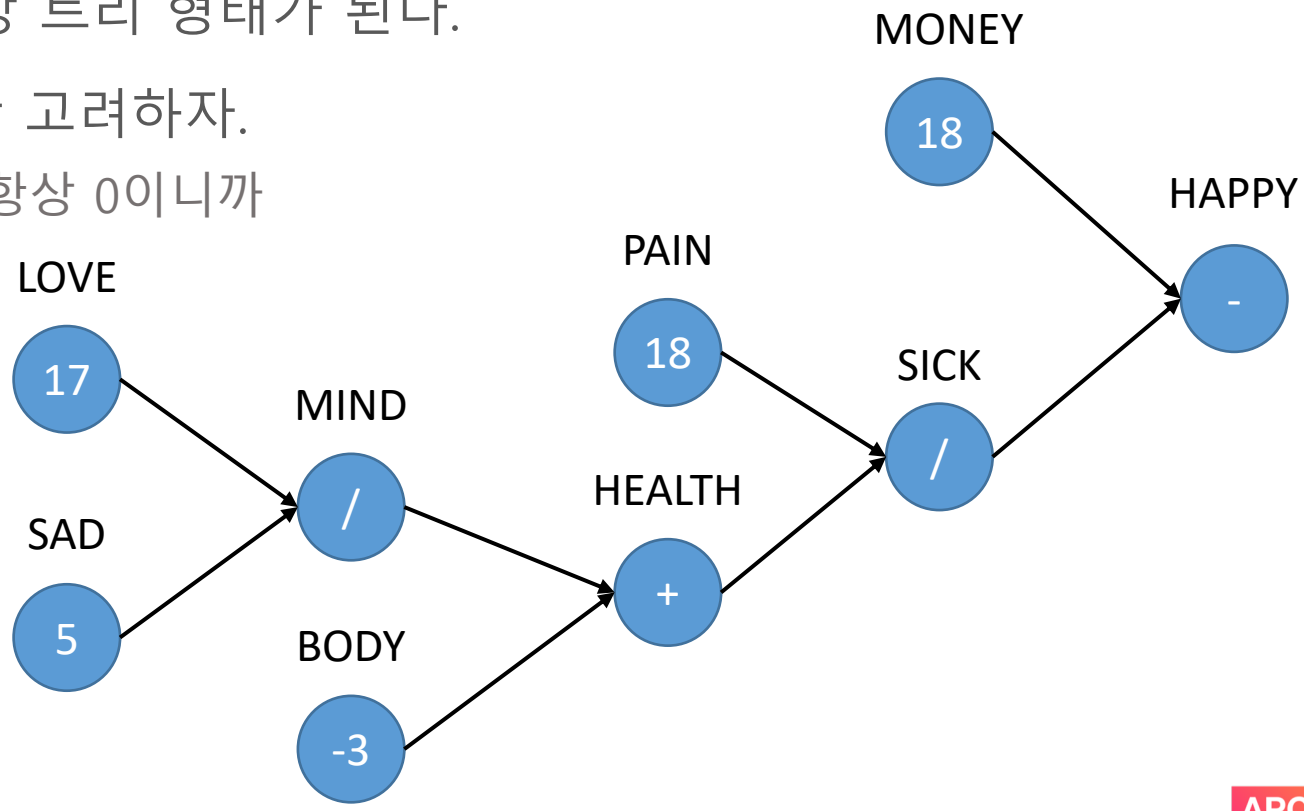
APC 2018

1H. 그날의 너

출제자: 김동이

환경적 요인은 변수로, 복합적 요인은 연산자로 계산 그래프(Computational Graph)를 그릴 수 있다.

- 문제의 조건에 의해서 그래프는 항상 트리 형태가 된다.
- HAPPY를 루트로 가지는 Sub-Tree만 고려하자.
  - 이외의 노드들은 어차피 영향력이 항상 0이니까

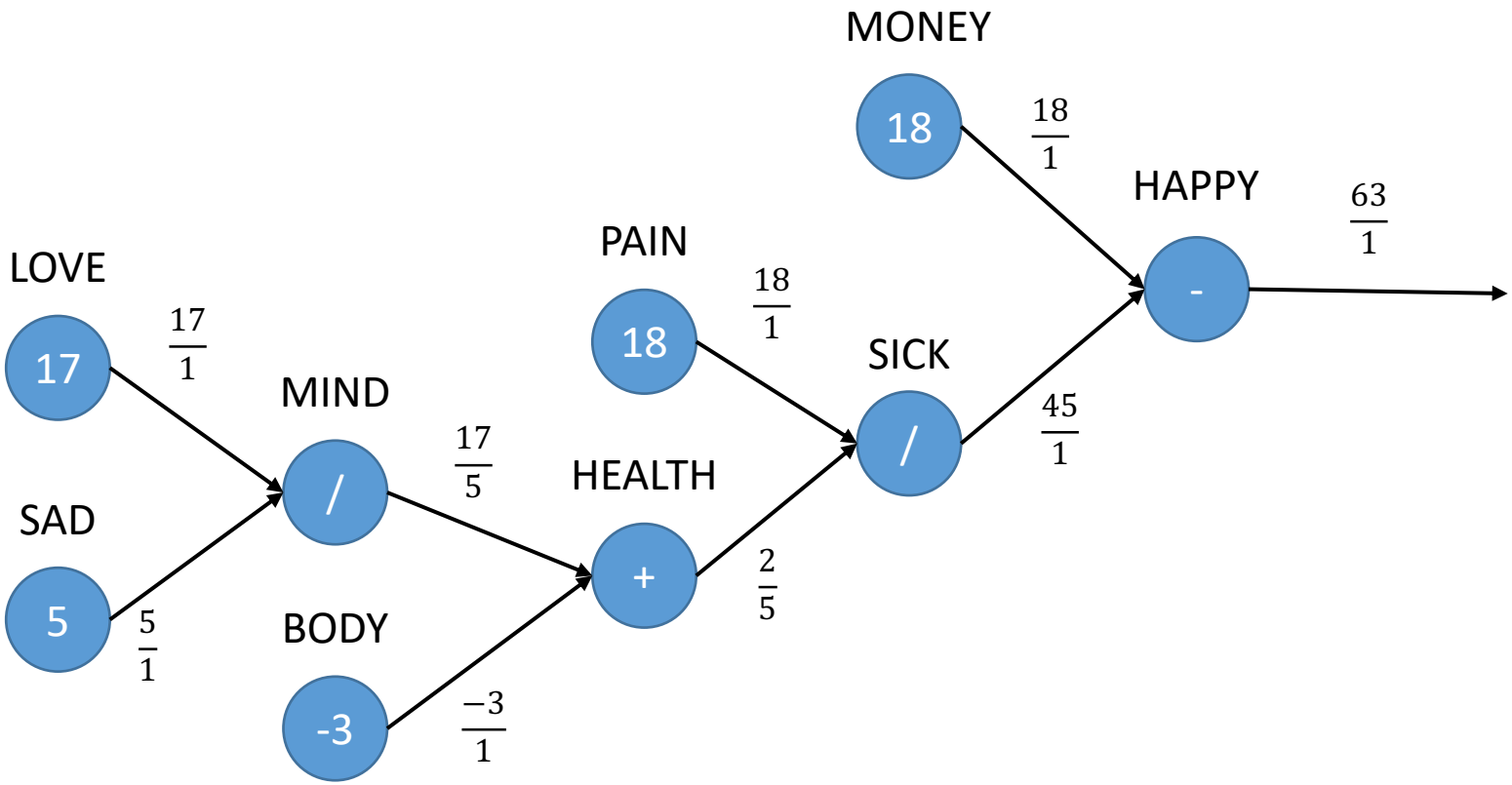


APC 2018

1H. 그날의 너

출제자: 김동이

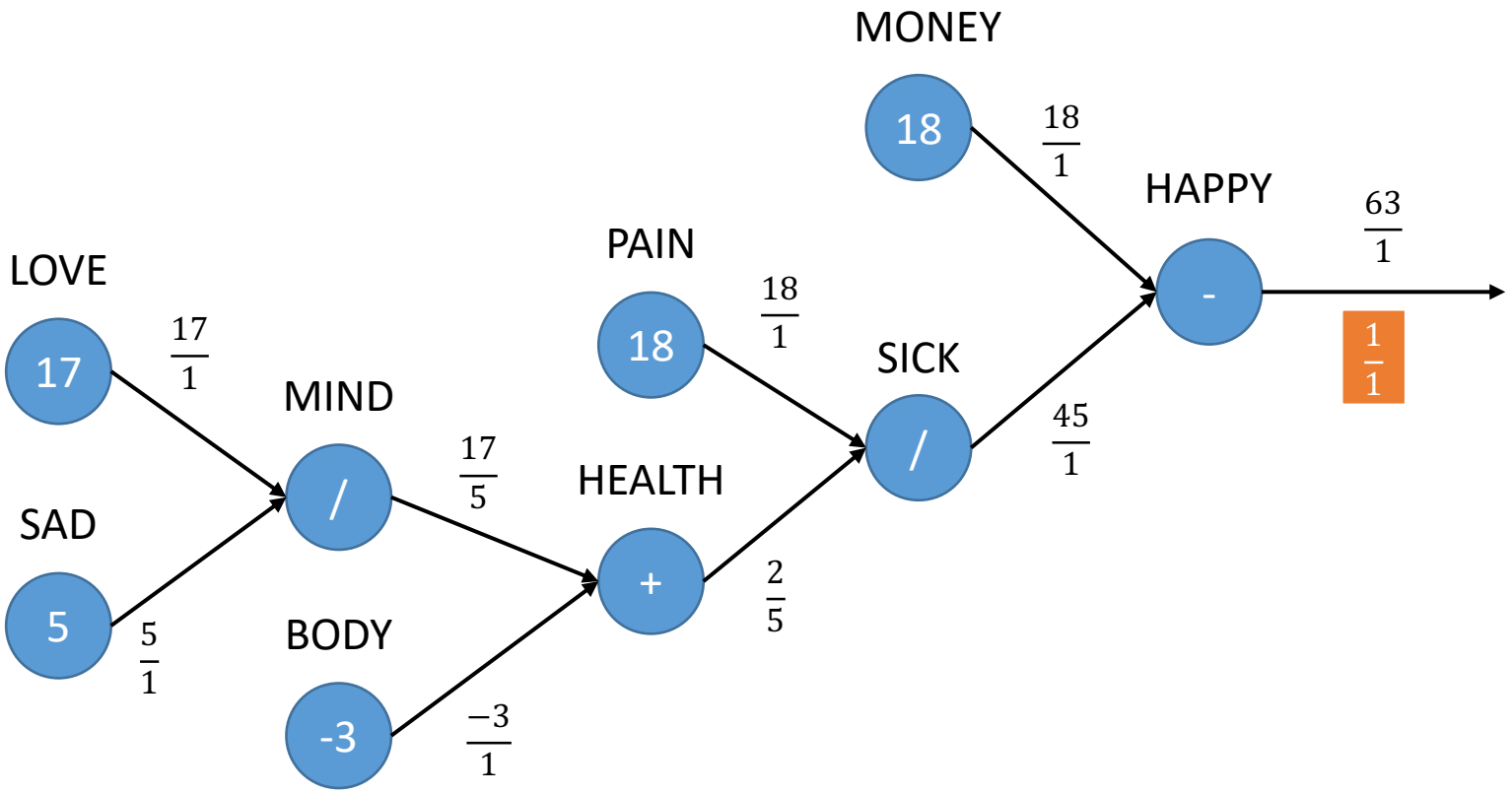
리프 노드들부터 위상 정렬을 이용해 각 요인들의 수치(출력)을 알 수 있다.



1H. 그날의 너

출제자: 김동이

일단 행복(HAPPY)의 영향력은 무조건 1이다.



1H. 그날의 너

출제자: 김동이

Chain Rule에 의해서 각 요인들은 아래 두 가지를 안다면 행복도에 대한 영향력을 계산할 수 있다.

- 부모(연산자)의 행복도에 대한 순간 변화율
- 자기 자신의 부모에 대한 순간 변화율

즉, 최고 조상 노드(HAPPY)의 영향력을 알고 있으므로, 역으로 계산해 나갈 수 있다.

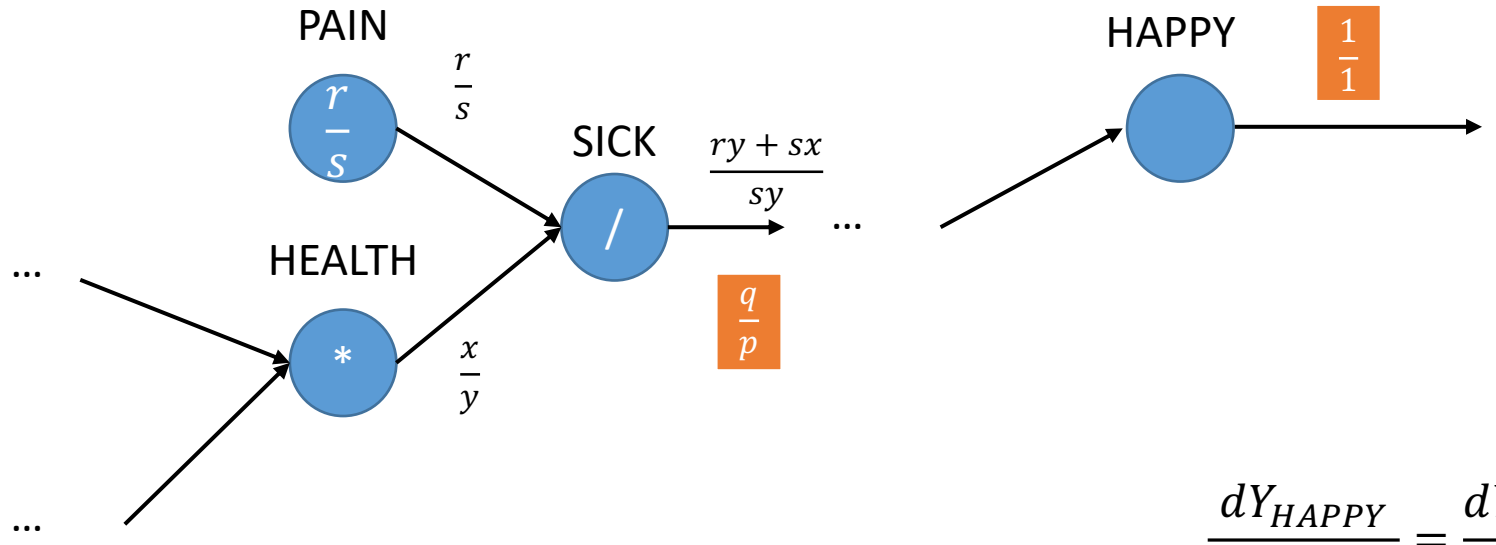
$$\frac{dy}{dx} = \frac{\dot{dy}}{du} \cdot \frac{du}{dx}$$

APC 2018

1H. 그날의 너

출제자: 김동이

각 연산자별로 Chain Rule을 이용해 루트부터 역으로 미분해 나간다.



$$\frac{dY_{HAPPY}}{dY_{HEALTH}} = \frac{dY_{HAPPY}}{dY_{SICK}} \cdot \frac{dY_{SICK}}{dY_{HEALTH}}$$

$$= \frac{q}{p} \cdot \{\frac{r}{s} \cdot (-1) \cdot \left(\frac{x}{y}\right)^{-2}\}$$

$$= -\frac{qry^2}{psx^2}$$

APC 2018

1H. 그날의 너

출제자: 김동이

요약하자면,

1. 위상 정렬을 사용해 리프 노드부터 각 요인의 수치를 계산해 나간다.
2. HAPPY 노드부터 역으로 Chain Rule로 편미분을 이용해 순간 변화율을 계산한다.

Deep Learning에서 사용되는 신경망의 Feed forward & Back propagation 기법이다.

APC 2018

2B. Router

출제자: 이주명

- 라우터의 버퍼는 큐라고 하는 자료구조의 형태이다. 큐를 구현하는 방법에 대해서는 아래 링크를 참고하자.
  - https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/
  - https://www.geeksforgeeks.org/queue-set-2-linked-list-implementation/
- 입력이 0보다 큰 경우는 enqueue 연산을 수행한다. 이때 큐에 들어있는 원소의 수가 버퍼의 크기와 같을 경우 버퍼가 꽉 찼다는 의미이므로 아무 연산도 수행하지 말아야 한다.
- 입력이 0인 경우는 dequeue 연산을 수행한다.
- 입력이 -1이면 큐에 아무것도 남지 않을 때까지 dequeue 연산을 수행하며 결과를 화면에 출력하면 된다.

2C. Hashing

출제자: 이주명

- 해시 값을 구하는 식은 문제에 나와있다. 이를 구하는 프로그램을 작성하면 된다.
- 주의할 점이 있는데, C언어에서 int형 변수의 크기는 4바이트이며 최대 $2^{31}-1=2,147,483,647$까지 표현할 수 있다. 즉, 모든 연산의 중간 결과가 이 범위를 넘어서는 안 된다는 것이다. 이보다 큰 long long형 변수의 경우 8바이트로 최대 $2^{63}-1=9,223,372,036,854,775,807$까지 표현이 가능하다. 하지만 문제의 식을 보면 나머지 연산을 하기 전 값이 최대가 되는 문자열은 50개의 z로 이루어져 있고, 그 값은 $26x(31^{50}-1)/30$이다. 이는 long long 변수로 표현할 수 있는 범위를 훨씬 넘는다.
- 하지만 만약 각 자릿수를 처리하면서 나머지 연산을 수행한다면 중간 결과로 나올 수 있는 값은 아무리 커봐야 $rM+26$이다. 그러므로 $h_1=a_{l-1}$, $h_i=(rh_{i-1}+a_{l-i}) \bmod M$ $(i>1)$이라는 수열을 정의하고 계산하면 $h_l=H$이 답이다.

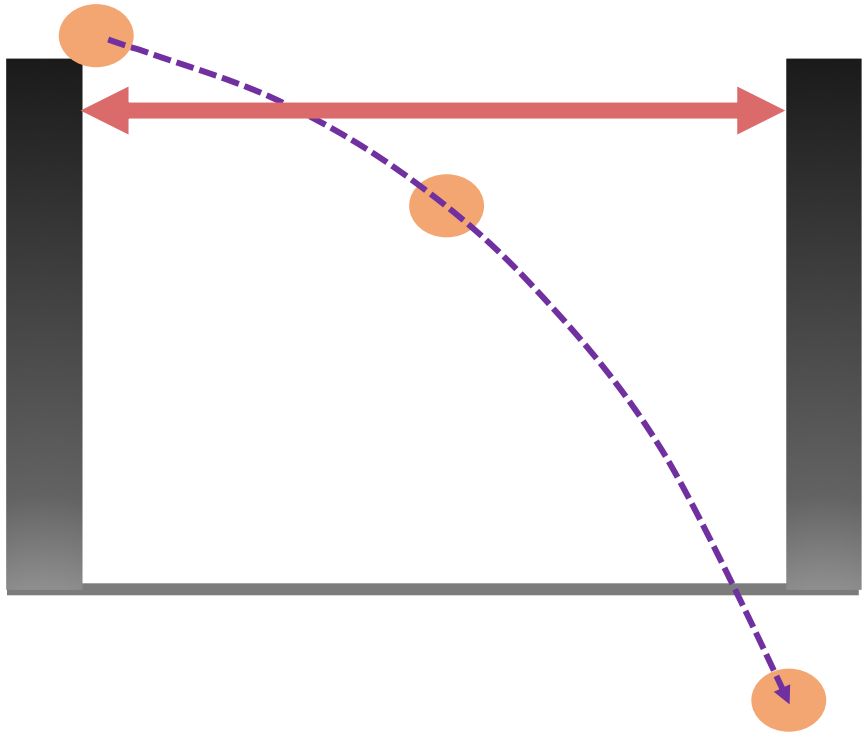
APC 2018

2D. 싱크홀

출제자: 홍준표

Small 부터 보자

- 한번만 부딪힌다면 ?



- 공기 저항이 없으므로 벽과 벽 사이를 이동하는 데 걸리는 시간 $t = W/V$
- 주어진 공식에 따라 돌이 수평방향으로 이동한 거리 $s = 5t^2$
- H보다 s가 크면 바닥에 먼저,
- 작으면 벽에 먼저 부딪힌다.
- ans = $H > S\ ?\ 1 : 0$

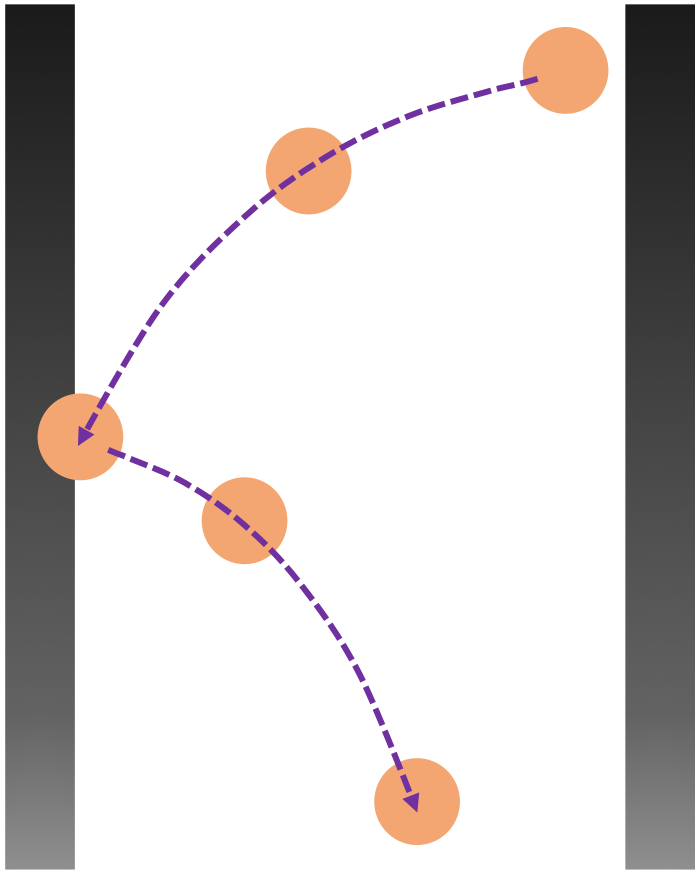
- 시간복잡도 : $O(1)$

APC 2018

2D. 싱크홀

출제자: 홍준표

- 한번으로 끝나지 않는다면 ?



- 문제의 조건에 따라 부딪히면 V 가 0.8 배가 된다.
- 새로 구해진 V 로 이동거리 s를 새로 구해 총 이동거리 D에 누적 시킨다.
- 총 이동거리 D 가 H 보다 커질 때 까지 위의 과정을 반복한다.
- 1회 반복은 곧 1회 충돌을 의미한다.
- t 는 V 에 반비례 하기 때문에 V 가 감소함에 따라 t 는 증가한다.
- 이동거리 s 는 t 가 증가함에 따라 기하급수적으로 증가한다.
- 따라서 총 이동거리 D 는 빠르게 H 에 도달한다.

- 시간복잡도 : $O(logH)$

APC 2018

2F. 준표의 조약돌

출제자: 김현정

- Small:
- [i, j] 구간의 흰 돌과 검은 돌의 개수를 어떻게 구할 수 있을까?

APC 2018

2F. 준표의 조약돌

출제자: 김현정

- Small:
- [i, j] 구간의 흰 돌과 검은 돌의 개수를 어떻게 구할 수 있을까?

- bsum[i] = i번째 돌까지 검은 돌의 개수
- bsum[i] = bsum[i – 1] + (dol[i] == 'B')
- 로 정의할때, [i, j] 구간의 검은 돌 개수: bsum[j] – bsum[i – 1]

2F. 준표의 조약돌

출제자: 김현정

- Small:
- [i, j] 구간의 흰 돌과 검은 돌의 개수를 어떻게 구할 수 있을까?

- bsum과 마찬가지로 흰 돌에 대한 wsum을 미리 계산해두면
- 구간 [i, j]에 대해 O(1)만에 흰 돌과 검은 돌의 개수를 구할 수 있다.
- 모든 구간 중 조건을 만족하는 (j – i + 1)이 가장 긴 길이가 답

2F. 준표의 조약돌

출제자: 김현정

- Small:
- 시간복잡도: $O(N^2)$
  - compute bsum, wsum: O(N)
  - search all [i, j]: $O(N^2)$

APC 2018

2F. 준표의 조약돌

출제자: 김현정

- Large:
- [i, j] 구간에 검은돌이 b개, 흰돌이 w개 있다면
- b <= B: 구간을 [i, j+1]로 확장
- b > B: 구간을 [i+1, j]로 축소
- 구간을 움직일때, 이번에 확장되거나 축소되는 돌만 확인하여 개수를 조정

- 위처럼 i, j를 이동하며 w >= W가 되는 구간 중
  (j – i + 1)의 길이의 최대값이 정답이 된다.

APC 2018

2F. 준표의 조약돌

출제자: 김현정

- Large:
- 시간복잡도: O(N)
  - i의 이동량: O(N)
  - j의 이동량: O(N)

APC 2018