강의 22: 동적 프로그래밍 IV

강의 개요

- 2가지 종류의 추측
- 피아노/기타 핑거링
- 테트리스 훈련
- 슈퍼 마리오 브라더스

Review:

* 동적 프로그래밍의 5단계

(a) 하위 문제 정의 — 하위 문제의 개수 세기
(b) 추측 (해법의 일부분) — 선택지의 개수 세기
(c) 하위 문제의 해법을 연관짓기 — 하위 문제당 시간 계산
(d) 재귀 + 기억 — 총 시간 = 하위 문제당 시간 · 하위 문제의 개수
또는 바닥부터 위로 동적 프로그래밍 표 만들기
하위 문제가 비순환적인지/위상학적 순서 확인

(e) 원래 문제 풀기: = 하위 문제이거나
하위 문제 해법들의 조합 ⇒ 추가적인 시간

* 2종류의 추측:

(A) (3)단계에서, 어떤 하위 문제를 사용할지 추측 (피보나치를 제외하고 모든 동적 프로그래밍에서 사용)
(B) (1)단계에서, 정답 구조에 대해 더 많은걸 기억하거나 추측하기 위해 하위 문제를 추가 가방 싸기 동적 프로그래밍에서 사용
- 하위 문제에게 해법을 효율적으로 알려줌
- 부모 하위 문제가 해법의 특징을 알 수 있게함

피아노/기타 핑거링:

피아노

[Parncutt, Sloboda, Clarke, Raekallio, Desain, 1997]

[Hart, Bosch, Tsai 2000]

[Al Kasimi, Nichols, Raphael 2007] etc.

- 주어진 음악적 조각, 즉 음 n개의 시퀀스를 오른손으로 연주
- 손가락 1,2,...,F = 5(사람의 경우)
- d(f,p,g,q) 음 p를 손가락 f로 연주한 다음 음 q를 손가락 g로 연주할 때의 난이도

  e.g., $1 < f < g$ & $p > q \Rightarrow$ 불편함
  잡아 당기는 규칙: $p \ll q \Rightarrow$ 불편함
  레가토(부드럽게) $\Rightarrow \infty$ ($f = g$일 때)
  약한 손가락 규칙: $g \in \{4,5\}$는 가급적 사용하지 않음
  $3 \to 4$ & $4 \to 3$ 귀찮음 ~ 기타등등.

첫 번째 시도:

1. ~~subproblem = min. difficulty for suffix notes[i:]~~
2. ~~guessing = finger f for first note[i]~~
3. recurrence:
   ~~$DP[i] = \min(DP[i+1] + d(\text{note}[i], f, \text{note}[i+1], ?)$ for $f \cdots$)~~
   → not enough information!

올바른 동적 프로그래밍:

1. 하위 문제 = i번째 음을 손가락 f로 연주할 때 suffix notes[i:]의 난이도 최소값
   ⇒ n·F subproblems

2. 추측 = 다음 i+1번째 음을 손가락 g로 연주함
   ⇒ F 개의 선택지

3. 반복: DP[i,f] = min(DP[i + 1,g] + d(note[i],f,note[i + 1],g)forg in range(F))
   DP[n,f] = 0
   ⇒ 하위 문제당 시간Θ(F)

4. 위상학적 순서: for i in reversed(range(n)):
   for f in 1,2,...,F:
   총 시간 $O(nF^2)$

5. 원래 문제 = min(DP[0,f]for f in 1,...,F)
   (가장 첫 번째 손가락을 추측)

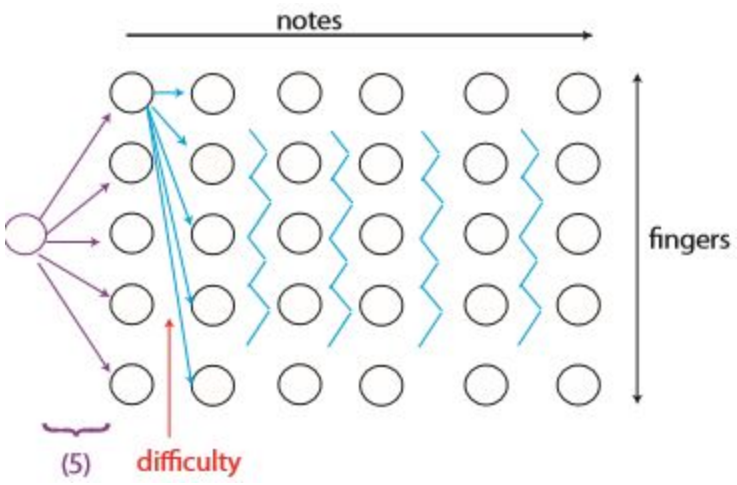

그림 1: DAG.

DAG로 계산할 시 각 간선에 비용을 부과하고,
최단 거리(가장 적은 비용)을 구한다.
해당 방법은 추측과 최솟값을 구하는 DP의 작동 원리와 똑같다.

기타

같은 음을 연주하는 최대 S 개의 방법! (S는 줄의 개수)

- "손가락"의 재정의 = 음을 연주하는 손가락 + 음을 연주하는 줄
- $\Rightarrow F \to F \cdot S$

일반화:

한 번에 여러 개의 음 e.g. 화음

- 입력: notes[i] = 최대 F개의 음들의 리스트
  (한 손가락으로 여러 음을 연주할 수 없음)
- 상태: 과거에 대해 알아야 함. F개의 손가락에서 F +1 (음 또는 아무것도 안 하는 상태)
  $\Rightarrow (F+1)^F$ 개의 대응

(1) $n \cdot (F+1)^F$ 개의 하위 문제, (F + 1)F 는 i번째 음을 연주하는 방법의 수
(2) $(F+1)^F$ 개의 선택 (i+1번째 음을 연주하는 방법)
(3) $n \cdot (F+1)^{2F}$ 의 총 시간

- 양손으로 칠 때도 적용, F = 10
- 적절한 d를 정의할 필요가 있음

DP문제는 기본적으로 하위 문제가 무엇인지 알아낸다는 점에서 비슷하다.

그림 2: 테트리스.

테트리스 훈련:

- n개의 테트리스 조각과 작은 폭 w의 빈 보드가 주어짐
- 회전 정도와 x 좌표를 결정해야 함
- 조각이 다른 조각 또는 바닥에 닿을때까지 떨어뜨림
- 꽉 찬 행은 지우지 않음
  위 두 인공 조건들 없이는 알 수 없음
  (하지만 비어있지 않은 보드와 큰 w의 경우 NP-완전)
- 목표: 살아남기. 예를 들어, 높이를 h 내에서 유지하기

첫 번째 시도:

1. ~~subproblem = survive in suffix i?~~ WRONG
2. guessing = how to drop piece i $\implies$ # choices = $O(w)$
3. ~~recurrence: $DP[i] = DP[i+1]$~~ ?! not enough information!
   What do we need to know about prefix : i?

정답:

- 1. 하위 문제= suffix i: 에서 생존?
  초기의 열이 채워진 정보인 $h_0, h_1, \cdots, h_{w-1}$ 가 주어진 경우 **h**라 명명
  $\Rightarrow$ # 하위 문제= O(n· h^w)

- 3. 반복: DP[i,**h**] = max(**h**의 조각 **i**의 유효한 이동 **m**에 대한 DP[i,**m**])
  $\Rightarrow$ 하위 문제의 시간 = O(w)

- 4. 위상학적 순서: for i in reversed(range(n)): for **h**···
  총 시간 = O(nwh^w) (위와 같은 DAG)

- 5. 해답 = DP[0,**0**]
  (그리고 이동을 복원하기 위해 부모 포인터 사용)

슈퍼 마리오 브라더스

플랫폼 비디오 게임

- 레벨 전체가 주어짐 (오브젝트, 적들, ...) (← n)
- w×h 의 작은 화면
- 게임 상태
  - 화면 이동 (← n)
  - 플레이어의 위치 & 속도 (O(1)) (← w)
  - 오브젝트의 상태, 몬스터의 위치, etc. (← $c^{w \cdot h}$)
  - 화면 밖의 모든 것은 초기 상태로 돌아감 (← $c^{w \cdot h}$)
  - 점수 (← S)
  - 시간 (← T)
- 전환 함수 δ: (게임 상태, 행동) → 게임 상태'
  아무것도 하지 않거나, ↑,↓,←,→, B, A 버튼을 누르고/떼는 행위

(1) 하위 문제: 게임 상태 C로부터의 최고의 점수 (또는 시간)
⇒ n· $c^{w \cdot h}$ ·S ·T 개의 하위 문제
(2) 추측: C에서 할 다음 행동
⇒ O(1) 선택지
(3) 반복:

$$DP(C) = \begin{cases} C.\text{score} & \text{if on flag} \\ \infty & \text{if } C.\text{dead or } C.\text{time} = 0 \\ \max(DP(\delta(C, A))) & \text{for } A \text{ in actions} \end{cases}$$

⇒ O(1) 하위 문제당 시간
(4) 위상학적 순서: 시간 오름차순
(5) 원래 문제: DP(초기의 게임 상태)

- S와 T의 유사 다항 시간
- n에 대해 다항 시간

- w와 h에 대해 지수적 시간