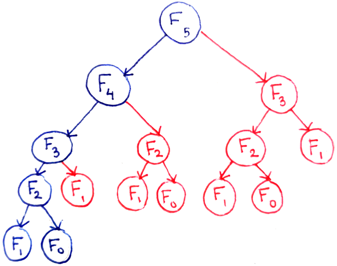
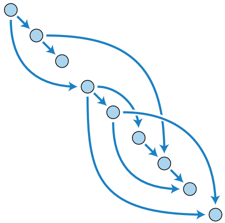
Dynamic Programming II
Partitions, Games, Dags, Search and Digits

Duarte Nóbrega

DCC - FCUP

Porto, November 19th, 2024





Overview

- Dynamic ... what?

Overview

- Dynamic ... what?
- Many concrete examples and live-coding

Overview

- Dynamic ... what?
- Many concrete examples and live-coding
    - Minimax Principle for Games
    - DP in DAGs (and Trees)
    - DP with Linear Partitions
    - DP with Bitmasks
    - Digit Dynamic Programming

What is Dynamic Programming?

- In very simple terms: **avoid computing** the same sub-problem over and over again!
- How?

What is Dynamic Programming?

- In very simple terms: **avoid computing** the same sub-problem over and over again!
- How? Trade-off between **Memory** $\longleftrightarrow$ **Time**
- Dynamic Programming (DP) ... the holy grail in the world of problem-solving techniques?

What is Dynamic Programming?

- In very simple terms: **avoid computing** the same sub-problem over and over again!
- How? Trade-off between **Memory** $\longleftrightarrow$ **Time**
- Dynamic Programming (DP) ... the holy grail in the world of problem-solving techniques?
- Not quite! Although a powerful technique requires **optimal substructure** and **overlapping sub-problems**.

How can I solve all DP problems?

Is there a standard way to approach all DP problems? Not really, but first devising a **recursive solution** helps!

Minimax Principle for Games

- How can we use DP to play games optimally?

Minimax Principle for Games

- How can we use DP to play games optimally?
    - Determine the **winning states**. If a player starts in such a state he will win *(providing he plays optimally)*
    - If it is a two-player game, it may be possible to model it as one player trying to **maximize** a global score while the other attempts to **minimize** it

Minimax Principle for Games

- How can we use DP to play games optimally?
  - Determine the **winning states**. If a player starts in such a state he will win *(providing he plays optimally)*
  - If it is a two-player game, it may be possible to model it as one player trying to **maximize** a global score while the other attempts to **minimize** it

Let's go over some examples ...

- ▸ [UVA] Bachet's Game
- ▸ [CSES] Removal Game

DP with Linear Partitions

- We want to **partition** an array a of size n into k disjoint consecutive sub-arrays that minimize or maximize a given cost function.

7	7	8	7	7	8	1	7

Figure: Array of size 8 partitioned into 3 sub-arrays.

DP with Linear Partitions

- We want to **partition** an array a of size n into k disjoint consecutive sub-arrays that minimize or maximize a given cost function.

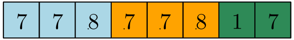


Figure: Array of size 8 partitioned into 3 sub-arrays.

Examples:

- ▸ [Codeforces] Bakery

DP with Linear Partitions

- We want to **partition** an array a of size n into k disjoint consecutive sub-arrays that minimize or maximize a given cost function.

7	7	8	7	7	8	1	7

Figure: Array of size 8 partitioned into 3 sub-arrays.

Examples:

- [Codeforces] Bakery

Follow-up: Can you solve the problem in $\mathcal{O}(n \times \log(n) \times k)$ time complexity?

DP in DAGs (and Trees)

- A Directed Acyclic Graph (DAG) is a **directed graph without cycles**

DP in DAGs (and Trees)

- A Directed Acyclic Graph (DAG) is a **directed graph without cycles**

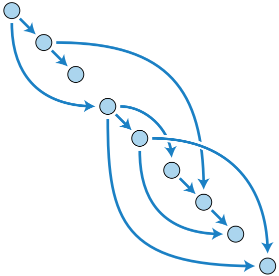


Figure: Sample DAG with 9 vertices.

DP in DAGs (and Trees)

- A Tree is a graph with **no cycles**

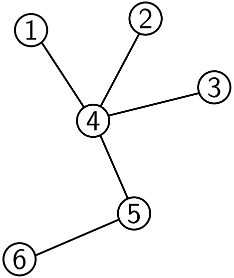


Figure: Tree with 6 vertices.

DP in DAGs (and Trees)

- A Tree is a graph with **no cycles**

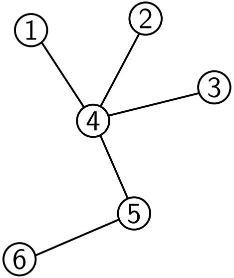


Figure: Tree with 6 vertices.

Examples:

- [AtCoder] Longest Path
- [AtCoder] Independent Set

DP with Bitmasks

- The Travelling Salesman Problem (TSP) asks for the **shortest** path that visits **every node** of a graph **exactly once**
- Some variants include finding the shortest length cycle

DP with Bitmasks

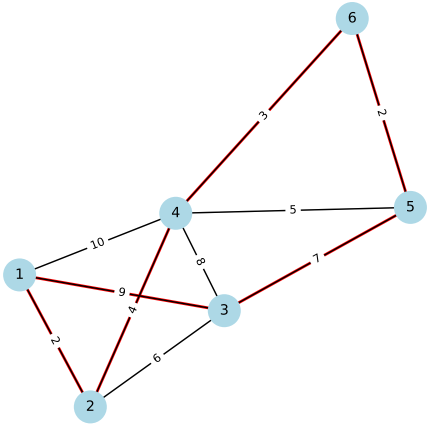


Figure: A sample (cyclic) tour in a graph $\mathcal{G}$ with 6 vertices having cost 27. *Optimal?*

DP with Bitmasks

- What is the state? ...

DP with Bitmasks

- What is the state? ...
- Use a *bitmask*! A *bitmask* can be seen as a sequence of $0's$ and $1's$. It can be used efficiently using the computer's binary representation of integers.

DP with Bitmasks

- What is the state? ...
- Use a *bitmask*! A *bitmask* can be seen as a sequence of $0's$ and $1's$. It can be used efficiently using the computer's binary representation of integers.
- Held–Karp Algorithm: $\mathcal{O}(n^2 \times 2^n)$ dynamic programming approach *(note the exponential factor!)*

DP with Bitmasks

- What is the state? ...
- Use a *bitmask*! A *bitmask* can be seen as a sequence of $0's$ and $1's$. It can be used efficiently using the computer's binary representation of integers.
- Held–Karp Algorithm: $\mathcal{O}(n^2 \times 2^n)$ dynamic programming approach *(note the exponential factor!)*

General TSP has been proved to be NP-Complete (the problem is hard, no polynomial solution is known ...)

It is APX-Hard (not even a polynomial time constant factor approximation algorithm is known ...)

DP with Bitmasks

- What is the state? ...
- Use a *bitmask*! A *bitmask* can be seen as a sequence of $0's$ and $1's$. It can be used efficiently using the computer's binary representation of integers.
- Held–Karp Algorithm: $\mathcal{O}(n^2 \times 2^n)$ dynamic programming approach *(note the exponential factor!)*

General TSP has been proved to be NP-Complete (the problem is hard, no polynomial solution is known ...)

It is APX-Hard (not even a polynomial time constant factor approximation algorithm is known ...)

Examples:

- [ED202] Procurando Pokemons

Digit Dynamic Programming

- How can we **efficiently** count numbers with a given property over a **large** $[L..R]$ range $(1 \leq L \leq R \leq 10^{18})$?
- ... what is the state?

Digit Dynamic Programming

- How can we **efficiently** count numbers with a given property over a **large** $[L..R]$ range $(1 \leq L \leq R \leq 10^{18})$?
- ... what is the state?
- Typically:
  - p - position of the digit being filled
  - $flag_{Upper}$ - is the number **lower** than the **upper limit**
  - $flag_{Bigger}$ - is the number **bigger** than the **lower limit**
  - ... other useful problem specific characteristics!
  - The code may be simplified by using the fact that: $count(L, R) = count(0, R) - count(0, L - 1)$
- Examples:
- ▸ [CSES] Counting Numbers
- ▸ [ONI'18] Problema A - Códigos preguiçosos

I Want to Know More!

Here you have a selection of additional resources that you may find useful:

- CP Algorithms: A good reference with high quality explanations
- Codeforces DP problems: A list of all DP tagged problems on Codeforces sorted by (expected) difficulty *(highly recommended!)*
- CSES DP Section: A well-crafted list of classical DP problems *(a good starting point)*
- AtCoder Educational DP contest: Curated list of 26 *essential* DP problems *(some require more advanced techniques that we did not cover in this course: matrix multiplication, convex hull trick ...)*