PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

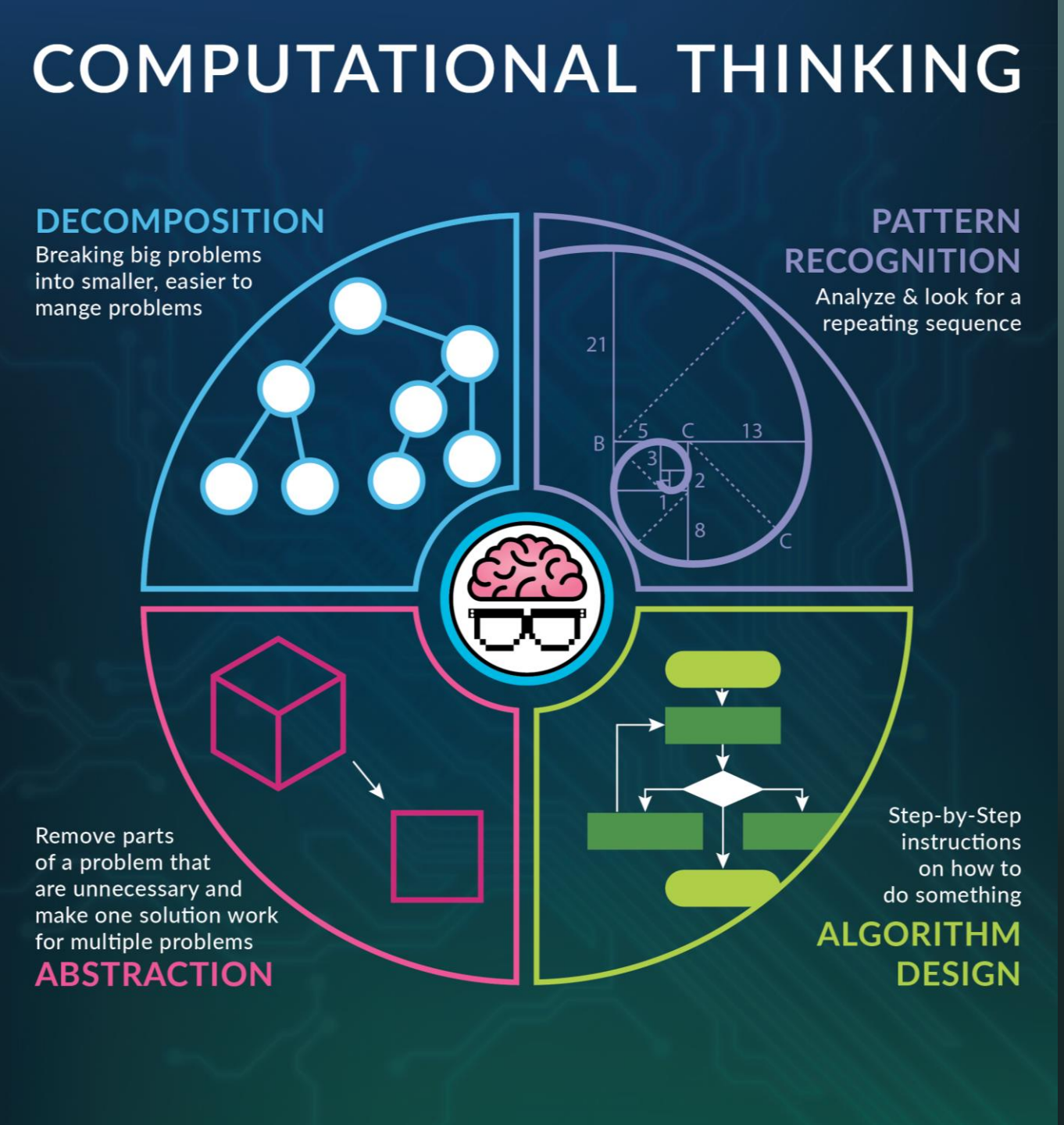
NHÓM 11:

ĐẶNG QUANG ANH TUẤN - 19522489

NGUYỄN HẢI ĐĂNG - 19521316

MỤC LỤC

- Nhắc lại về Computational Thinking.
- Bài tập vận dụng.



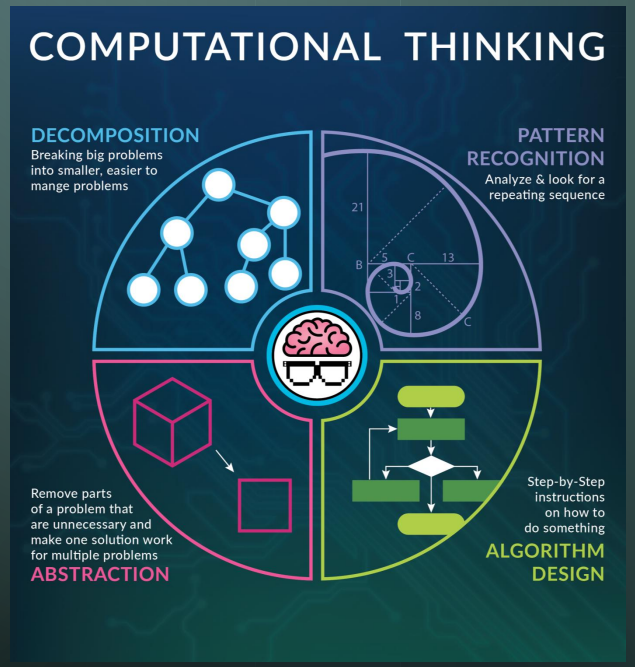
Computational Thinking

- **-Khái niệm: Tư duy tính toán là quá trình tiếp cận một vấn đề một cách có hệ thống tạo ra và thể hiện một giải pháp sao cho có thể thực hiện được bằng máy tính.**

- **Các kỹ thuật:**
  - **Decomposition.**
  - **Pattern recognition**
  - **Abstraction.**
  - **Algorithm Design.**

Decomposition:

- Tách vấn đề lớn thành những mảng nhỏ
- Đơn giản hóa nó một cách dễ hiểu
- Cụ thể đến mức khả thi để giải quyết
- Chia thành từng cụm có mối quan hệ với nhau.



Abstaction:

- Loại bỏ những phần không cần thiết
- Tạo ra giải pháp làm việc với một nhóm vấn đề

Pattern Recognition

- Xác định những điểm tương đồng và khác biệt
- Tìm kiếm sự liên kết, sự lặp lại và xác định chu kỳ
- Tổng hợp lại thành một mô thức tổng quát.

Algorithm Design

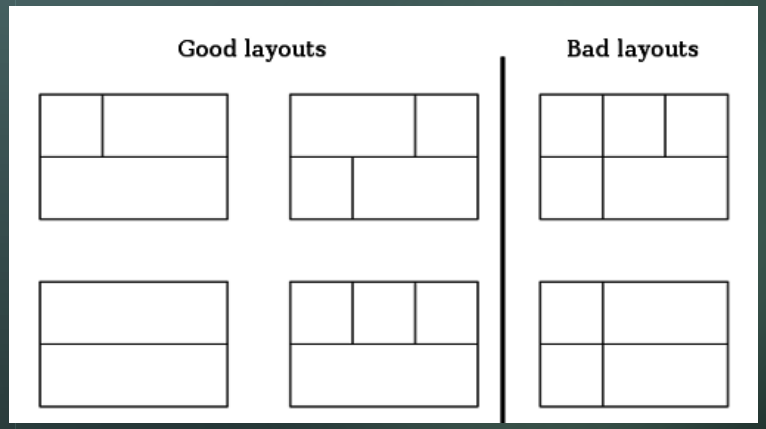
- Tạo ra các bước có thứ tự để giải quyết vấn đề
- Tuân thủ đúng theo trình tự và phải có kết thúc, không nhảy vọt
- Rõ ràng và ranh giới của từng bước, không mập mờ.

- Bài toán:

Bạn có vô số 4 loại khối lego có kích thước như

(chiều sâu x chiều cao x chiều rộng):

d	h	w
1	1	1
1	1	2
1	1	3
1	1	4



- Sử dụng các khối này, bạn muốn tạo một bức tường có chiều cao N và chiều rộng M. Đặc điểm của tường là:
  - Tường không được có bất kỳ lỗ nào trên đó.
  - Bức tường bạn xây nên là một kết cấu vững chắc, không nên có một vết đứt dọc thẳng trên tất cả các hàng gạch.
  - Các viên gạch phải được đặt theo chiều ngang.
- Có bao nhiêu cách xây tường? Trả về số cách xếp với module (10^9 + 7)
- Ràng buộc: $1 \leq N, M \leq 1000$

Các bước phân tích: Phân tích theo mô hình Computational Thinking.

Abstraction:

Decomposition:

Pattern Recognition:

Algorithm Design:

Link: (code có bộ test).

Các bước giải quyết :

1. Abstraction:

- Cho một bức tường có chiều cao N và chiều rộng M. Tính số cách sắp xếp toàn bộ trừ đi số cách sắp xếp sai

2. Decomposition:

Sau khi Abstaction thì ta thấy có thể chia bài toán thành 2 bài toán con

- (i). Tính tất cả số cách xếp.(4 loại thành 1 hàng ngang)
  + Tính số cách xếp trên từng hàng. Sau đó tính cách xếp toàn bộ dựa trên các hàng đã tính.
- (ii). Tính số cách xếp sai.(Có một vết nứt dọc trên tất cả hàng gạch)

=> Đáp án: (1) - (2)

3. Pattern recognition:

- Bài toán đếm: cụ thể là đếm số cách xếp 4 viên gạch xếp vào một bức tường sao cho thỏa mãn yêu cầu đề bài.

4. Algorithm Design:

Từ những bài toán con mà ta đã decomposition dẫn đến phương pháp thiết kế như này:

- Gọi f[i] là số cách chọn mảng có kích thước 1 * i.
- Số cách chọn hình có kích thước N * i là: g[i] = f[i]^n.
- Giả sử ta đã đặt được khối (1 x k) thì phần còn lại cần đặt có độ dài (i - k) vì vậy phần còn lại sẽ có cách chọn là f[i−k] .

Do đó ta có công thức:

$$f[i] = \begin{cases} f[i-1] + f[i-2] + f[i-3] + f[i-4] & \text{if } i > 0 \\ 1 & \text{if } i = 0 \\ 0 & \text{if } i < 0 \end{cases}.$$

- Ta có g[M] là số cách chọn với kích thước N * M, và để loại bỏ các trường hợp không chắc chắn:
  - Giả sử h[i] là số cách chọn thỏa yêu cầu bài toán với kích thước n*i.
  - Số cách chọn để xây tường không chắc chắn kích thước n*i là temp = $\sum_{j=1}^{i-1} h[j] * g[i-j]$
  - Khi đó số cách chọn theo yêu cầu đề ra với kích thước n*i là:
  - h[i] = g[i] - temp
- Vậy số cách chọn thỏa mãn đề bài là: H[m].

5. Độ phức tạp thuật toán: O(nlogn + m^2).

6. Programing.

Link:

https://colab.research.google.com/drive/1BjLoWSEc9mBTUVVoMFBfFLutFPGGHEuI#scrollTo=UDgZkI72mm1F

Bài toán:

Thám tử Rust đang điều tra một vụ giết người và anh ta muốn truy đuổi kẻ sát nhân. Kẻ sát nhân biết rằng mình chắc chắn sẽ bị bắt nếu đi đường chính để bỏ trốn, vì vậy hắn sử dụng đường làng để chạy khỏi hiện trường vụ án.

Rust biết rằng kẻ sát nhân sẽ chiếm đường làng và anh ta muốn đuổi theo hắn. Anh ta đang quan sát bản đồ thành phố, nhưng nó không hiển thị đường làng trên đó và chỉ hiển thị những con đường chính.

Bản đồ của thành phố là một biểu đồ bao gồm N nút (được dán nhãn từ 1 đến N) trong đó một nút cụ thể S thể hiện vị trí hiện tại của Rust và phần còn lại của các nút biểu thị các địa điểm khác trong thành phố và cạnh giữa hai nút là một đường chính giữa hai nơi trong thành phố. Có thể giả định một cách hợp lý rằng một cạnh không tồn tại / không được hiển thị trên bản đồ là đường làng. Điều đó có nghĩa là, có một đường làng giữa hai nút a và b khi không có đường thành phố giữa chúng.

Trong bài toán này, khoảng cách được tính bằng số đường làng giữa hai địa điểm bất kỳ trong thành phố.
Hãy giúp Rust tính khoảng cách ngắn nhất từ vị trí của anh ta (Nút S) đến tất cả các địa điểm khác trong thành phố nếu anh ta chỉ di chuyển bằng đường làng.

Dữ liệu:
• Dòng đầu tiên nhập 2 số nguyên, N thành phố, M con đường.

• M dòng tiếp theo nhập 2 số nguyên x, y biểu thị con đường giữa 2 thành phố x và y.

• Dòng cuối cùng nhập S, vị trí của Rust.

Ràng buộc:

- $2 \le N \le 2 * 10^5$
- $0 \le M \le 120000$
- $1 \le x,y,S \le N$

Kết quả:
in một dòng duy nhất bao gồm N-1 số nguyên được phân tách bằng dấu cách, biểu thị khoảng cách ngắn nhất của N-1 vị trí còn lại từ vị trí của Rust (ngoài trừ vị trí S) theo thứ tự tăng dần của số đỉnh

Ví dụ:

Input	Output
4 3 1 2 2 3 1 4 1	3 1 2

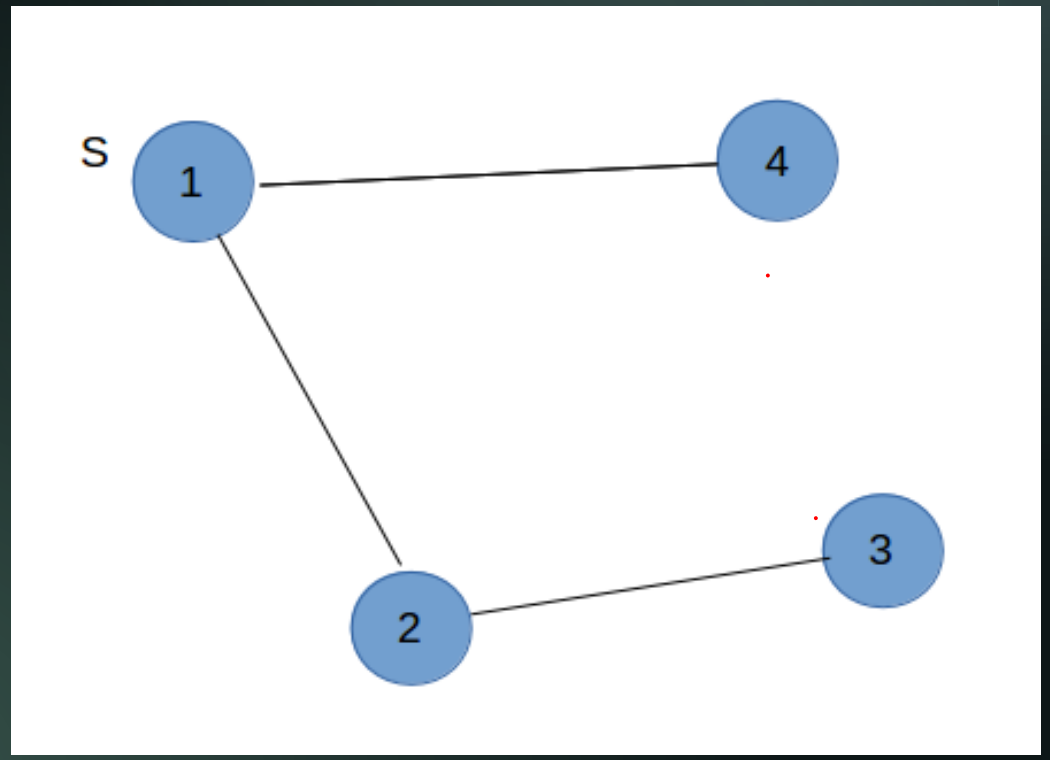
Giải thích:

Đầu vào :

4 3
1 2
2 3
1 4
1

Đầu ra :

3 1 2



Giải thích:

- Khoảng cách từ 1->2: 1-> 3-> 4-> 2, kết quả là 3.
- Khoảng cách từ 1->3: 1-> 3, kết quả là 1.
- Khoảng cách từ 1->4: 1->3->4, kết quả là 2.

Các bước phân tích: Phân tích theo mô hình Computational Thinking.

Abstraction:

Decomposition:

Pattern Recognition:

Algorithm Design:

Link: (code có bộ test).

Các bước giải quyết

1.Abstraction

- Cho đồ thị vô hướng N đỉnh và M cạnh. Tính khoảng cách ngắn nhất từ vị trí S đến tất cả đỉnh còn lại sao cho không đi qua các cạnh của M.

2.Decomposition

Sau khi Abstraction thì ta có thể chia bài toán thành 2 bài toán con đó là:

- Tính khoảng cách từ S đến tất cả đỉnh không có cạnh kết nối với S
- Tính khoảng cách từ S đến tất cả các đỉnh có cạnh kết nối với S

3.Pattern Recognition

- Đồ thị vô hướng.
- BFS

4.Algorithm designed

- Từ những mối quan hệ mà ta đã decomposition dẫn đến phương pháp thiết kế thuật toán như này:
  - Bước 1: Tính khoảng cách từ S đến tất cả các đỉnh không có cạnh kết nối với S
  - Bước 2: Khởi tạo tập U là những đỉnh có cạnh kết nối với S
  - Bước 3: Khởi tạo tập to_collect(T_i) để lưu trữ những đỉnh không có cạnh kết nối với S
  - Bước 4: Xét U_i có cạnh kết nối với T hay không. Nếu không có thì $d(U_i)$ sẽ bằng $d(T_i) + 1$, rồi xóa U_i ra khỏi U

5.Đánh giá độ phức tạp thuật toán

- Độ phức tạp: O(V + E)
  - V: số đỉnh của đồ thị
  - E: số cạnh của đồ thị

6. Programing.

- Link:

https://colab.research.google.com/drive/1f28SEyh8TJCMfK0uL3YwEk0QeSMGOiV-#scrollTo=rp9FYyG3jKgK

- Đề bài:

Thám tử Rust trong một lần truy đuổi kẻ sát nhân nhưng không may đã để lộ ra tin tức nơi ở của mình, cuối cùng lại bị kẻ sát nhân truy đuổi. Trong quá trình bị truy sát Rust nhìn thấy được thành phố, nhưng không may trước thành phố lại có một mê cung phải thông qua mê cung mới được vào thành phố. Rust biết nếu mình có thể vào thành phố thì sẽ sống, ngược lại sẽ chết. Vì vậy chỉ còn cách tiến vào mê cung.

Mê cung được cho minh họa dưới dạng ma trận nhị phân N * N, trong đó khối nguồn là khối trên cùng bên trái, tức là maze[0][0] và khối đích là khối dưới cùng bên phải, tức là maze[N-1][N-1]. Trong ma trận mê cung, 0 có nghĩa là ngõ cụt và 1 là khối có thể sử dụng làm đường đi.

Một lần nữa hãy giúp Rust tìm lối thoát của mê cung .In ra "DEATH" nếu mê cung đó không có lối ra, ngược lại in ra “ALIVE" và đường đi .

Dữ liệu: Nhập vào sô nguyên N, kích thước ma trân

Di chuyển theo 4 hướng: lên xuống trái phải

+ 2 ≤ N ≤ 50

Giả sử mê cung được cho như này:	Đường đi:
1 1 1 1 1	1 1 1 0 0
0 0 1 0 1	0 0 1 0 0
1 0 1 1 1	0 0 1 0 0
1 1 1 0 0	0 0 1 0 0
0 0 1 1 1	0 0 1 1 1

Các bước phân tích: Phân tích theo mô hình Computational Thinking.

Abstraction:

Decomposition:

Pattern Recognition:

Algorithm Design:

Link: (code có bộ test).

Các bước giải quyết:

1. **Abstraction.**

Cho ma trận (maze) nhị phân N * N.

Tìm đường đi là các con số 1 từ maze[0,0] đến maze[N-1,N-1].Nếu không có in ra "DEATH", ngược lại in ra "ALIVE" và đường đi

2. Decomposition

- Từ bài toán tìm đường dài ta chia thành tìm những đường ngắn hơn

3. **Pattern Recognition**

- Backtracking
- Recursion

4. Algorithm Design.

Từ mối quan hệ đó là tìm những đường đi ngắn, dẫn đến phương pháp tiếp cận sau.

Phương pháp tiếp cận: Hình thành một hàm đệ quy, hàm này sẽ đi theo 1 con đường và kiểm tra có đến đích hay không. Nếu không đến đích thì quay lại chọn con đường khác

- Tạo ra một ma trận kết quả khởi tạo bằng 0.
- Tạo ra hàm lấy ma trận đầu vào, ma trận kết quả và vị trí xét (i, j).
- Nếu vị trị không trong ma trận đầu vào hay không hợp lệ thì quay lại bước trước.
- Đánh dấu vị trị là 1 ở ma trận đầu ra và kiểm tra nếu là đích thì in ma trận kết quả.
- Gọi ví trí (i+1, j) và (I, j+1).
- Gán vị trí ở ma trận kết quả là 0.

5. Độ phức tạp của thuật toán.

O(4^{n^2}).

6. Programing.

Link:

Maze.ipynb - Colaboratory (google.com)


Thank
for
watching