**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**======\*======**

****

**BÁO CÁO BTL THUỘC HỌC PHẦN:**

**THỰC TẬP CƠ SỞ NGÀNH**

**NGHIÊN CỨU BÀI TOÁN**

**PHÂN CÔNG CÔNG VIỆC**

|  |  |  |
| --- | --- | --- |
| GVHD: | Nguyễn Thị Mỹ Bình | |
| Nhóm - Lớp: | 11- 20222IT6040002 | |
| Thành viên: | Nguyễn Đình Hội | 2022603030 |
|  | Lê Thị Ngọc Lan | 2022602329 |
|  | Trần Song Hào | 2022602338 |
|  | Nguyễn Tiến Hiền | 2022602039 |
|  | Bùi Duy Khánh | 2022601170 |

Hà Nội - 2023

[LỜI MỞ ĐẦU 2](#_Toc179446668)

[CHƯƠNG 1. Cơ sở lý thuyết 4](#_Toc179446669)

[1.1. Cơ sở lý thuyết trong bài toán phân công công việc 4](#_Toc179446670)

[1.1.1. Phát biểu bài toán 4](#_Toc179446672)

[1.1.2. Ứng dụng bài toán 4](#_Toc179446673)

[1.2. Cơ sở lý thuyết của các thuật toán 5](#_Toc179446674)

[1.2.1. Cơ sở lý thuyết của thuật toán tham lam 5](#_Toc179446675)

[1.2.1.1. Lý thuyết chung về giải thuật tham lam 5](#_Toc179446676)

[1.2.1.2. Nguyên lý hoạt động của thuật toán tham lam 6](#_Toc179446678)

1.2.1.3. Các thành phần và mô hình chung của thuật toán tham lam

[1.2.2. Cơ sở lý thuyết của thuật toán nhánh cận 7](#_Toc179446680)

[1.2.2.1. Lý thuyết chung về thuật toán nhánh cận 7](#_Toc179446681)

[1.2.2.2. Nguyên lý hoạt động của thuật toán nhánh cận 8](#_Toc179446683)

1.2.2.3. Các thành phần và mô hình chung của thuật toán nhánh cận

[CHƯƠNG 2. Thiết kế thuật toán để giải bài toán phân công công việc 11](#_Toc179446685)

[2.1. Thiết kế thuật toán tham lam ( Greedy) 11](#_Toc179446686)

2.1.1. Thuật toán tham lam trong bài toán phân công công việc

2.1.2. Các bước giải thuật

[2.2. Thiết kế thuật toán nhánh cận ( Branch and Bound) 11](#_Toc179446687)

2.2.1. Thuật toán nhánh cận trong bài toán phân công công việc

2.2.2. Các bước giải thuật

[2.3. Đánh giá hiệu quả các phương pháp 11](#_Toc179446688)

[CHƯƠNG 3. Cài đặt và kiểm thử 12](#_Toc179446689)

[3.1. Cài đặt thuật toán 12](#_Toc179446690)

[3.1.1. Cài đặt thuật toán tham lam 12](#_Toc179446691)

[3.1.2. Cài đặt thuật toán nhánh cận 12](#_Toc179446692)

[3.2. Kiểm thử 12](#_Toc179446693)

[KẾT LUẬN 12](#_Toc179446694)

[TÀI LIỆU THAM KHẢO 12](#_Toc179446695)

# LỜI MỞ ĐẦU

Hiện nay, trong bối cảnh nền công nghiệp hoá hiện đại hoá ngày càng phát triển, việc quản lý lực lượng lao động là một vấn đề cần phải quan tâm. Tỉ lệ thất nghiệp của thanh niên năm 2023 tại Việt Nam là 13%, tương đương với 64,9 triệu người, là mức thấp nhất trong 15 năm qua. Để giảm được tỉ lệ thất nghiệp, chúng ta phải giải quyết vấn đề cốt lõi là phân công công việc. Vậy nên, việc quản lí và phân công lực lượng, công việc là một phần không thể thiếu trong lao động sản xuất.

Bài toán phân công công việc (Job Assignment Problem) là một dạng bài toán tối ưu hoá. Bài toán thường được sử dụng để phân công một nhóm người thực hiện một số công việc nhất định sao cho tổng chi phí thực hiện các công việc là nhỏ nhất ( thường chi phí đó có thể là thời gian, năng suất, …).

Chúng em mong muốn đáp ứng được nhu cầu của người dùng: tối ưu hoá lực lượng lao động, trang thiết bị, nguyên vật liệu, ngoài ra còn giảm chi phí, thời gian sản xuất dẫn tới tăng hiệu suất công việc. Và đề tài này ứng dụng rất nhiều trong đời sống, quản lý và dịch vụ. Nó khả thi trong việc áp dụng vào mọi lĩnh vực, ngành nghề và có thể nhanh chóng đưa ra cách giải quyết vấn đề. Ví dụ như phân công công việc cho các công nhân trong dây chuyền sản xuất để tối ưu hoá năng suất, hay là phân công nhân viên vào các nhiệm vụ khác nhau trong một dự án để hoàn thành dự án nhanh nhất có thể, …

Hiểu được tầm quan trọng của việc phân công công việc, chúng em đã quyết định chọn đề tài nghiên cứu là: “Nghiên cứu cơ sở lý thuyết, ứng dụng và cài đặt ít nhất 2 thuật toán để giải bài toán Phân công công việc” để nghiên cứu và phát triển. Hai thuật toán chúng em lựa chọn là thuật toán Tham lam và thuật toán Nhánh cận. Cả 2 thuật toán đều là những thuật toán điển hình để giải những bài toán tối ưu hoá, giúp chúng em có thể giải được bài toán đã đề ra.

Chúng em xin gửi lời cảm ơn chân thành đến cô Nguyễn Thị Mỹ Bình đã tâm huyết giúp đỡ, hướng dẫn chúng em trong quá trình học tập học phần Thực tập cơ sở ngành. Cô đã giúp chúng em tích luỹ được nhiều kiến thức để có thể hoàn thành được bài báo cáo đề tài này.

Trong quá trình thực hiện bài báo cáo, do hiểu biết của chúng em còn hạn chế, khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được những lời góp ý của thầy cô để bài báo cáo ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

# Cơ sở lý thuyết

## Cơ sở lý thuyết trong bài toán phân công công việc

### Mở đầu

Bài toán phân công công việc là một bài toán tối ưu hóa thuộc lĩnh vực toán học tổ hợp, trong đó chúng ta cần phân công một nhóm công nhân để hoàn thành một nhóm công việc sao cho tổng chi phí thực hiện công việc là nhỏ nhất. Bài toán này có ý nghĩa quan trọng trong nhiều lĩnh vực, bao gồm sản xuất, quản lý tài nguyên và các hệ thống phân bổ nhân sự.

Bài toán phân công công việc đã được nghiên cứu từ rất lâu và vẫn là một trong những bài toán quan trọng có nhiều ứng dụng trong thực tế, đặc biệt là trong các lĩnh vực như logistics, lập lịch trình, và tối ưu hoá nguồn lực.

### Phát biểu bài toán

Cho n công việc và n công nhân, trong đó công việc thứ i cần được thực hiện với một chi phí Cij khi được phân công cho công nhân j. Nhiệm vụ của bạn là phân công mỗi công nhân một công việc sao cho tổng chi phí là nhỏ nhất.

* Input:
  + Dòng đầu tiên chứa một số nguyên dương n.
  + n dòng tiếp theo, mỗi dòng chứa n số nguyên là ma trận chi phí Cij.
* Output:
* Dòng đầu tiên in ra tổng chi phí nhỏ nhất.
* Dòng thứ hai in ra thứ tự phân công công việc theo chỉ số của các công nhân.

### Ý tưởng

Bài toán này có thể được giải quyết bằng nhiều phương pháp khác nhau, bao gồm:

* Thuật toán vét cạn (Brute Force): Thử tất cả các khả năng phân công công việc và chọn phương án tối ưu. Tuy nhiên, phương pháp này chỉ khả thi với bài toán có kích thước nhỏ do độ phức tạp tính toán tăng theo cấp số nhân, với tổng số khả năng là n!.
* **Thuật toán nhánh cận** (Branch and Bound): Phương pháp này giúp giảm số lượng trường hợp cần duyệt bằng cách cắt bớt những nhánh không dẫn tới nghiệm tối ưu. Cụ thể, cây tìm kiếm được chia thành nhiều nhánh, và các nhánh có chi phí lớn hơn giới hạn cận trên sẽ bị loại bỏ ngay từ đầu.
* **Thuật toán tham lam** (Greedy Algorithm) là một phương pháp giải bài toán trong đó các quyết định được đưa ra dựa trên việc chọn lựa tối ưu trong từng bước, mà không xem xét đến các quyết định trong tương lai. Ý tưởng chính của thuật toán tham lam là xây dựng một giải pháp theo từng bước, mỗi bước được chọn là tốt nhất trong số các lựa chọn có sẵn tại thời điểm đó.

## Cơ sở lý thuyết của các thuật toán

**1.2.1.1 Lý thuyết chung về giải thuật tham lam**

- Thuật toán tham lam là một cách tiếp cận để giải quyết vấn đề, đưa ra một loạt các lựa chọn, từng lựa chọn một, với mục tiêu đạt được giải pháp tối ưu. Ở mỗi bước, thuật toán tham lam sẽ chọn tùy chọn khả dụng tốt nhất dựa trên một số tiêu chí được xác định trước, mà không xem xét bối cảnh toàn cục hoặc hậu quả tiềm ẩn của lựa chọn trong các bước tiếp theo. Nguyên tắc chính là luôn đưa ra lựa chọn tối ưu cục bộ, hy vọng rằng hiệu ứng tích lũy của những lựa chọn này sẽ dẫn đến giải pháp tổng thể tốt nhất.

- Trong phương pháp tham lam, việc lựa chọn quyết định tối ưu được thực hiện dựa trên thông tin hiện có mà không lo lắng về tác động mà những quyết định này có thể gây ra trong tương lai. Các thuật toán tham lam dễ phát minh, dễ triển khai và hầu hết thời gian đều khá hiệu quả

**1.2.1.2 Nguyên lý hoạt dộng của thuật toán tham lam**

Giải thuật tham lam xây dựng các giải pháp bằng cách lựa chọn hành động tốt nhất tại mỗi bước, mà không xem xét tác động của hành động đó đến các bước sau. Ý tưởng này dựa trên quan sát rằng, nếu mỗi lần chọn hành động tốt nhất tại thời điểm hiện tại, ta có thể đạt được kết quả tối ưu toàn cục.

**1.2.1.3 Các thành phần của giải thuật tham lam**

* Một tập hợp các ứng viên (candidate), để từ đó tạo ra lời giải
* Một hàm lựa chọn, để theo đó lựa chọn ứng viên tốt nhất để bổ sung vào lời giải
* Một hàm khả thi (feasibility), dùng để quyết định nếu một ứng viên có thể được dùng để xây dựng lời giải
* Một hàm mục tiêu, ấn định giá trị của lời giải hoặc một lời giải chưa hoàn chỉnh
* Một hàm đánh giá, chỉ ra khi nào ta tìm ra một lời giải hoàn chỉnh.

1.2.1.4 Mô hình chung



### Cơ sở lý thuyết của thuật toán nhánh cận

#### Lý thuyết chung về thuật toán nhánh cận

* Thuật toán nhánh cận (Branch and Bound) là một phương pháp giải quyết các bài toán tối ưu tổ hợp, trong đó tìm kiếm được tổ chức dưới dạng cây. Mỗi nhánh của cây đại diện cho một tập các quyết định và được cắt bớt khi phát hiện ra không thể tìm được lời giải tốt hơn trong nhánh đó. Ý tưởng chính của thuật toán nhánh cận là phân chia bài toán lớn thành các bài toán con nhỏ hơn và lần lượt khám phá các lời giải tiềm năng trong các nhánh này.
* Ứng dụng của Thuật Toán Nhánh Cận:

- Tối ưu hóa tổ hợp: Thuật toán nhánh cận được sử dụng rộng rãi trong việc giải quyết các bài toán tối ưu hóa tổ hợp như bài toán người bán hàng, bài toán cái túi và phân công công việc.

- Vấn đề thỏa mãn ràng buộc: Thuật toán có khả năng xử lý hiệu quả các vấn đề thỏa mãn ràng buộc bằng cách khám phá có hệ thống không gian tìm kiếm và cắt tỉa các nhánh dựa trên các ràng buộc.

- Phân bổ tài nguyên: Thuật toán được áp dụng trong các kịch bản như phân bổ tài nguyên, nơi cần phân phối tài nguyên một cách tối ưu giữa các nhu cầu cạnh tranh.

* Ưu điểm của Thuật Toán Nhánh Cận:

-Tính tối ưu: Thuật toán đảm bảo tính tối ưu trong các giải pháp cho những bài toán thỏa mãn các điều kiện nhất định, đảm bảo rằng giải pháp tốt nhất có thể được tìm thấy.

- Hiệu quả bộ nhớ: Thuật toán thường yêu cầu ít bộ nhớ hơn so với các phương pháp tìm kiếm toàn diện khác như brute force, đặc biệt đối với các bài toán có không gian tìm kiếm lớn.

- Tính linh hoạt: Thuật toán có thể điều chỉnh cho nhiều lĩnh vực vấn đề khác nhau và có thể đáp ứng các biểu diễn và ràng buộc vấn đề khác nhau.

- Song song hóa: Các thuật toán nhánh cận có thể được song song hóa hiệu quả, cho phép khám phá nhanh chóng không gian tìm kiếm bằng cách sử dụng nhiều bộ xử lý hoặc tài nguyên tính toán.

* Nhược điểm của Thuật Toán Nhánh Cận:

- Độ phức tạp Việc triển khai các thuật toán nhánh cận có thể rất phức tạp, đặc biệt đối với các bài toán có ràng buộc phức tạp và không gian tìm kiếm lớn.

- Phụ thuộc vào Heuristic: Hiệu quả của thuật toán nhánh cận phụ thuộc nặng nề vào chất lượng của hàm giới hạn (bounding function) và các heuristics được sử dụng để hướng dẫn tìm kiếm, điều này không phải lúc nào cũng có sẵn hoặc dễ dàng thiết kế.

- Khó khăn trong môi trường động: Thuật toán không phù hợp cho các môi trường động hoặc thay đổi, nơi mà các ràng buộc hoặc mục tiêu của bài toán có thể thay đổi thường xuyên, vì nó dựa vào một phiên bản bài toán cố định để khám phá không gian tìm kiếm.

#### Nguyên lý hoạt động của thuật toán nhánh cận

Thuật toán sử dụng cây tìm kiếm, mỗi nút đại diện cho một trạng thái phân công công việc, mỗi nhánh đại diện cho việc lựa chọn một công việc được giao cho một người cụ thể.

* **Nhánh (Branch):** Tại mỗi cấp độ của cây, bạn chọn một công việc và thử phân công nó cho một người khác nhau, tạo ra các nhánh con. Các nhánh con sẽ tiếp tục phân công những công việc còn lại cho đến khi tất cả các công việc đều được gán.
* **Cận (Bound):** Để giảm thiểu số lượng nhánh phải xét, tại mỗi nút, ta tính một giới hạn dưới cho tổng thời gian tối thiểu có thể đạt được. Nếu giới hạn này lớn hơn hoặc bằng tổng thời gian của nghiệm tốt nhất đã biết, ta có thể cắt bỏ nhánh đó mà không cần tiếp tục xét.

#### Các thành phần và mô hình của thuật toán nhánh cận

Phương pháp nhánh và cận là một dạng cải tiến của phương pháp quay lui, được áp dụng để tìm nghiệm của bài toán tối ưu. Giả sử nghiệm của bài toán có thể biểu diễn dưới dạng một vector ( x1, x2, …., xn) mỗi thành phần xi ( i = 1,2, . . , n) được chọn ra từ tập Si. Mỗi nghiệm của bài toán X = ( x1, x2 ,…., xn) được xác định “ độ tốt” bằng một hàm f (X) và mục tiêu cần tìm nghiệm có giá trị f (X) đạt giá trị nhỏ nhất (hoặc đạt giá trị lớn nhất). Tư tưởng của phương pháp nhánh và cận như sau: Giả sử, đã xây dựng được k thành phần ( x1, x2, …., xk) của nghiệm và khi mở rộng nghiệm ( x1, x2, …., xk+1) , nếu biết rằng tất cả các nghiệm mở rộng của nó ( x1, x2, …., xk+1, …..) nếu không tốt bằng nghiệm tốt nhất đã biết ở thời điểm đó, thì ta không cần mở rộng từ ( x1, x2, …., xk) nữa. Như vậy, với phương pháp nhánh và cận, ta không phải duyệt toàn bộ các phương án để tìm ra nghiệm tốt nhất mà bằng cách đánh giá các nghiệm mở rộng, ta có thể cắt bỏ đi những phương án (nhánh) không cần thiết, do đó việc tìm nghiệm tối ưu sẽ nhanh hơn. Cái khó nhất trong việc áp dụng phương pháp nhánh và cận là đánh giá được các nghiệm mở rộng, nếu đánh giá được tốt sẽ giúp bỏ qua được nhiều phương án không cần thiết, khi đó thuật toán nhánh cận sẽ chạy nhanh hơn nhiều so với thuật toán vét cạn.

Thuật toán nhánh cận có thể mô tả bằng mô hình đệ quy sau:

A diagram of a flowchart

Description automatically generated

# Thiết kế thuật toán để giải bài toán phân công công việc

## Thiết kế thuật toán tham lam ( Greedy)

### Thuật toán tham lam trong bài toán phân công công việc

* Mục tiêu của bài toán là phân công n công việc cho n công nhân sao cho mỗi công nhân chỉ làm một công việc và mỗi công việc chỉ do một công nhân thực hiện. Tổng thời gian thực hiện các công việc là nhỏ nhất.
* Phương pháp tham lam đưa ra các lựa chọn tốt nhất ở mỗi bước (theo thời gian nhỏ nhất), mà không xét đến tác động lâu dài của những lựa chọn đó.
* Tư tưởng tham lam trong bài toán này là:

Tại mỗi bước, chọn công việc có thời gian ngắn nhất cho một công nhân mà công việc đó chưa được thực hiện bởi bất kỳ công nhân nào khác.

### Các bước giải thuật

#### Bài toán và phân tích yêu cầu

* **Input**:
  + Một ma trận cost[n][n], trong đó:
    - cost[i][j] đại diện cho thời gian mà công nhân thứ i cần để thực hiện công việc thứ j.
  + Số lượng công nhân là n.
* **Output**:
  + Một phương án phân công công việc cho các công nhân sao cho mỗi công nhân được gán một công việc, mỗi công việc chỉ được thực hiện bởi một công nhân, và tổng thời gian là nhỏ nhất.
  + In ra tổng thời gian tối thiểu.
* **Hạn chế**: Mỗi công nhân chỉ làm một công việc và mỗi công việc chỉ do một công nhân thực hiện.

#### Giải thuật

Bước 1: Khởi tạo bài toán

* Đầu vào: Chúng ta có một tập hợp các lựa chọn (ở đây là tập hợp các công việc) và một tập các đối tượng (công nhân) cần gán.
* Biến khởi tạo: Mảng để lưu các công việc đã được gán, tất cả ban đầu đều chưa được gán (giá trị false).
* Một mảng để ghi nhận công việc nào được gán cho từng đối tượng.
* Một biến để lưu giá trị tối ưu cần tính

Bước 2: Xác định lựa chọn cục bộ

* Lựa chọn cục bộ tối ưu là lựa chọn tại mỗi bước, công việc nào cần được gán cho công nhân hiện tại.
* Đối với bài toán này, lựa chọn cục bộ là tìm công việc có thời gian ngắn nhất chưa được gán cho công nhân hiện tại.
* Điều kiện tham lam: Mỗi bước chọn công việc tốt nhất mà không quan tâm đến các bước sau.

Bước 3: Cập nhật trạng thái bài toán

* Sau khi chọn công việc tốt nhất, công việc đó được gán cho công nhân hiện tại, và được đánh dấu là "đã gán".
* Cập nhật chi phí hoặc thời gian vào tổng chi phí.

Bước 4: Lặp lại quá trình

* Tiếp tục thực hiện lựa chọn cục bộ (chọn công việc tốt nhất) cho công nhân tiếp theo và cập nhật trạng thái cho đến khi tất cả công nhân đều được gán công việc.

Bước 5: Kết thúc và xuất kết quả:

* Sau khi tất cả các công nhân đều được gán công việc, kết thúc quá trình lặp và xuất ra kết quả (công việc đã gán cho mỗi công nhân và tổng thời gian thực hiện).

#### Lưu đồ thuật toán



#### Độ phức tạp thời gian

* Vòng lặp chính: Duyệt qua tất cả các công nhân (tức là n công nhân).
* Tìm công việc tốt nhất cho mỗi công nhân: Với mỗi công nhân, phải duyệt qua tất cả các công việc (tức là n công việc).

=> Do đó, độ phức tạp thời gian của thuật toán là **O(n²)**, với n là số công nhân (và số công việc).

## Thiết kế thuật toán nhánh cận ( Branch and Bound)

### Thiết kế thuật toán nhánh cận

#### Bài toán và phân tích yêu cầu

Input:

- n (số công nhân, cũng là số công việc): Số nguyên dương n >= 1

- Ma trận thời gian tịj Ma trận kích thước n x n, trong đó phần tử tịj là thời gian công nhân thứ *i* (với (*i* = 1, 2, ..., n) cần để hoàn thành công việc *j* (với *j* = 1, 2, ..., n).

Output:

- Phân công tối ưu: Danh sách các cặp công nhân và công việc sao cho mỗi công nhân chỉ làm một công việc và mỗi công việc chỉ do một công nhân thực hiện.

- Tổng thời gian nhỏ nhất (TTG): Tổng thời gian tối thiểu để hoàn thành tất cả các công việc theo phương án phân công tối ưu.

Hạn chế:

1. Số lượng công nhân và công việc phải bằng nhau: Đảm bảo n công nhân và n công việc.

2. Mỗi công nhân chỉ có thể làm một công việc: Không có công nhân nào có thể thực hiện nhiều công việc cùng lúc.

3. Mỗi công việc chỉ được làm bởi một công nhân: Không được có công việc nào bị trùng lặp.

4. Giới hạn kích thước ma trận: Khi n lớn, bài toán sẽ có độ phức tạp cao (thuộc lớp bài toán NP-Complete), gây khó khăn trong việc tìm ra giải pháp tối ưu một cách nhanh chóng.

5. Thời gian thực hiện từng công việc tịj phải là số dương: Không thể có thời gian âm.

#### Các bước giải thuật

1. Khởi tạo

* **Tạo trạng thái gốc**: Ở trạng thái này, chưa có công việc nào được phân công. Đây là gốc của cây tìm kiếm.

1. Phát triển nhánh

* Tại mỗi cấp độ của cây tìm kiếm, chọn một công việc và thử phân công công việc đó cho một người.
* Sau khi phân công một công việc cho một người, ta tạo ra các nhánh con tương ứng, trong đó mỗi nhánh đại diện cho việc phân công công việc tiếp theo cho các người còn lại.

1. Tính toán tổng thời gian và cận dưới

* **Tính tổng thời gian tạm thời**: Sau mỗi bước phân công, tính tổng thời gian của các công việc đã được phân công.
* **Tính cận dưới** 
  + Để tối ưu hóa quá trình tìm kiếm, tính giới hạn cận dưới cho thời gian tối thiểu có thể đạt được.
  + Cận dưới có thể được tính bằng cách cộng tổng thời gian tạm thời với thời gian nhỏ nhất có thể cho các công việc chưa được phân công.
  + Cận dưới giúp loại bỏ những nhánh không khả thi.

1. Cắt nhánh

* Sau khi tính được tổng thời gian và cận dưới cho một nhánh:
  + So sánh cận dưới với giá trị của nghiệm tốt nhất hiện tại (nếu có).
  + Nếu cận dưới của nhánh đó lớn hơn hoặc bằng tổng thời gian của nghiệm tốt nhất hiện tại, cắt nhánh đó và không mở rộng thêm nhánh con từ nhánh này.
  + Nếu tổng thời gian tạm thời nhỏ hơn nghiệm tốt nhất, tiếp tục phân nhánh để tìm các phương án tốt hơn.

1. Cập nhật nghiệm tối ưu

Khi một trạng thái phân công đầy đủ được tìm thấy (tất cả công việc đều được phân cho một người):

* + So sánh tổng thời gian của trạng thái này với nghiệm tốt nhất hiện tại.
  + Nếu tổng thời gian của trạng thái này nhỏ hơn, cập nhật nghiệm tốt nhất bằng trạng thái hiện tại.

1. Tiếp tục tìm kiếm hoặc dừng lại

* Lặp lại quá trình phát triển nhánh, tính cận dưới và cắt nhánh cho đến khi: Tất cả các nhánh khả thi đều được xét hoặc cắt bỏ.
* Khi không còn nhánh khả thi nào để mở rộng, nghiệm tối ưu sẽ là nghiệm cuối cùng được cập nhật.

#### Thiết kế giải thuật

A screenshot of a computer screen

Description automatically generated

## Đánh giá hiệu quả các phương pháp

### Phương pháp sử dụng thuật toán tham lam (Greedy)

#### Ưu điểm:

* **Đơn giản và dễ hiểu:** Thuật toán dựa trên cách tiếp cận tham lam (greedy), luôn chọn công việc có thời gian nhỏ nhất cho mỗi công nhân tại mỗi bước. Điều này giúp thuật toán dễ viết và dễ triển khai.
* **Thời gian chạy chấp nhận được:** Với độ phức tạp là O(n²) do vòng lặp tìm công việc có thời gian nhỏ nhất và gán nó cho công nhân, thuật toán có thể hoạt động tốt với số lượng công nhân và công việc không quá lớn.

#### Nhược điểm:

* **Không tối ưu toàn cục:** Mặc dù thuật toán này luôn tìm công việc nhanh nhất cho mỗi công nhân tại từng bước, nhưng nó có thể không cho ra giải pháp tối ưu toàn cục. Trong một số trường hợp, việc chọn một công việc khác (không phải công việc có thời gian nhỏ nhất) có thể giúp giảm tổng thời gian. Đó là hạn chế của các thuật toán tham lam.
* **Phụ thuộc vào cách chọn:** Thuật toán này không thử nhiều lựa chọn khác nhau mà chỉ theo một chiến lược duy nhất. Điều này có thể bỏ lỡ các trường hợp tối ưu hơn.

#### Tính khả thi:

* Với bài toán này, nếu số công nhân và công việc bằng nhau và không có công việc nào bị bỏ trống, thuật toán sẽ luôn hoàn thành với một phương án hợp lệ. Tuy nhiên, trong trường hợp số lượng công nhân hoặc công việc thay đổi, bạn sẽ cần điều chỉnh thuật toán để đảm bảo tính hợp lệ của bài toán.

# Cài đặt và kiểm thử

## Cài đặt thuật toán

### Cài đặt thuật toán tham lam

### Cài đặt thuật toán nhánh cận

## Kiểm thử

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO