MỤC LỤC

[**1.** **Định nghĩa.** 2](#_Toc153117661)

[1.1. Cấu trúc liên kết. 2](#_Toc153117662)

[1.2. Nút (node). 2](#_Toc153117663)

[1.3. ArrayList. 2](#_Toc153117664)

[1.4. HashMap (Bảng băm). 2](#_Toc153117665)

[1.5. Tree (Cây). 2](#_Toc153117666)

[1.6. Cây tìm kiếm nhị phân. 3](#_Toc153117667)

[1.7. Đồ thị. 3](#_Toc153117668)

[1.8. Đỉnh (Vertex). 3](#_Toc153117669)

[1.9. Cạnh (Edge). 3](#_Toc153117670)

[1.10. Đường đi 3](#_Toc153117671)

[1.11. Thành phần liên thông. 3](#_Toc153117672)

[1.12. Đường găng: 4](#_Toc153117673)

[1.13. Thuật toán đường găng. 4](#_Toc153117674)

[*2.* **Bài toán đặt ra.** 4](#_Toc153117675)

[2.1. Bài toán 4](#_Toc153117676)

[2.2. Vấn đề 4](#_Toc153117677)

[2.3. Tính cần thiết. 4](#_Toc153117678)

[*3.* **Các cấu trúc dữ liệu.** 4](#_Toc153117679)

[3.1. ArrayList 4](#_Toc153117680)

[*3.1.1.* Giới thiệu 4](#_Toc153117681)

[*3.1.2.* Mục đích sử dụng 5](#_Toc153117682)

[3.2. Đồ thị 5](#_Toc153117683)

[*3.2.1.* Giới thiệu 5](#_Toc153117684)

[*3.2.2.* Mục đích sử dụng 5](#_Toc153117685)

[3.3. Hash Map 5](#_Toc153117686)

[*3.3.1.* Giới thiệu 5](#_Toc153117687)

[*3.3.2.* Mục đích sử dụng. 5](#_Toc153117688)

[3.4. Binary Searching Tree 5](#_Toc153117689)

[*3.4.1.* Giới thiệu 5](#_Toc153117690)

[*3.4.2.* Mục đích sử dụng. 5](#_Toc153117691)

[*4.* **Thuật toán sử dụng.** 5](#_Toc153117692)

[4.1. Tìm kiếm tuyến tính. 5](#_Toc153117693)

[4.2. Tìm kiếm nhị phân 5](#_Toc153117694)

[*5.* **Các đối tượng** 6](#_Toc153117695)

[*6.* **Lộ trình phát triển sau môn học.** 6](#_Toc153117696)

[6.1. Sử dụng không gian mạng 6](#_Toc153117697)

[6.2. Đa nền tảng 6](#_Toc153117698)

[6.3. Trí tuệ nhân tạo trong việc tự động phân bổ nguồn lực, thời gian. 6](#_Toc153117699)

1. **Định nghĩa.**
   1. Cấu trúc liên kết.

Cấu trúc liên kết thường liên quan đến cách các phần tử dữ liệu được tổ chức và kết hợp với nhau trong một chương trình hoặc một cấu trúc dữ liệu.

* 1. Nút (node).

Trong cấu trúc liên kết, nút "node" là một thành phần cơ bản hoặc một đơn vị cơ bản. Mỗi node chứa thông tin hoặc dữ liệu và một liên kết đến node tiếp theo trong danh sách liên kết. Cấu trúc của một node thường bao gồm hai thành phần chính:

+ Dữ liệu (Data): Phần này lưu trữ thông tin hoặc giá trị mà node đại diện.

+ Liên kết (Link hoặc Tham chiếu): Phần này chứa địa chỉ hoặc tham chiếu đến node tiếp theo trong danh sách liên kết.

* 1. ArrayList.

ArrayList là một lớp trong Java, thuộc gói java.util, được sử dụng để tạo một mảng có thể thay đổi kích thước (dynamic array). Nó là một dạng cấu trúc dữ liệu danh sách mảng (List) và thực chất là một mảng đối tượng có thể mở rộng mà không cần phải khai báo kích thước cố định từ trước.

Các điểm chính về ArrayList bao gồm:

* **Kích thước động**: ArrayList có thể thay đổi kích thước theo nhu cầu, tự động điều chỉnh khi các phần tử được thêm vào hoặc loại bỏ.
* **Truy cập ngẫu nhiên**: Bạn có thể truy cập bất kỳ phần tử nào trong ArrayList bằng cách sử dụng chỉ mục, giống như mảng thông thường.
  1. HashMap (Bảng băm).

HashMap là một Cấu trúc dữ liệu đặc biệt, cho phép người dùng truy vấn đặc biệt với độ phức tạp thời gian là O(1), nó cung cấp một cấu trúc dữ liệu khóa-giá trị (key-value) dựa trên bảng băm. Nó được sử dụng để lưu trữ và quản lý dữ liệu theo cặp khóa-giá trị.

Các điểm chính về HashMap bao gồm:

* **Khóa không trùng lặp**: Mỗi khóa trong HashMap phải là duy nhất. Nếu bạn thêm một giá trị với một khóa đã tồn tại, giá trị cũ sẽ bị ghi đè bởi giá trị mới.
* **Null cho cả khóa và giá trị**: HashMap cho phép sử dụng null cho cả khóa và giá trị. Điều này có nghĩa là bạn có thể có một hoặc cả hai đều là null.
* **Không duy trì thứ tự**: HashMap không đảm bảo thứ tự của các phần tử, nghĩa là không có sự sắp xếp cố định của các cặp khóa-giá trị.
* **Hiệu suất cao**: HashMap được thiết kế để cung cấp hiệu suất tốt trong việc thêm, tra cứu và xóa phần tử.
  1. Tree (Cây).

Cây là một cấu trúc dữ liệu phân cấp, gồm các nút liên kết với nhau theo cách sao cho một nút có thể có nhiều nhất là một nút cha và nhiều nhất là n nút con, trong đó n có thể là bất kỳ số nguyên không âm nào. Cấu trúc dữ liệu cây thường được sử dụng để mô hình hóa các quan hệ phân cấp giữa các phần tử dữ liệu.

Một số thuật ngữ chính liên quan đến cây:

* **Nút (Node):** Đại diện cho một phần tử trong cây. Mỗi nút có thể có một hoặc nhiều nút con.
* **Gốc (Root):** Nút không có nút cha, đó là nút ở đỉnh cùng của cây.
* **Nút con (Child):** Nút mà có một nút cha.
* **Nút lá (Leaf):** Nút không có nút con.
* **Nút cha (Parent)**: Nút mà có tham chiếu đến một nút con khác.
* **Mức (Level):** Mức của một nút là khoảng cách từ nút đó đến gốc. Gốc có mức là 0.
* **Dạng cây (Shape of Tree):** Các cấu trúc của cây, bao gồm cả sự sắp xếp của nút và các mức của chúng.
* **Chiều cao (Height):** Chiều cao của một cây là mức lớn nhất của nút trong cây.
  1. Cây tìm kiếm nhị phân.

Cây tìm kiếm nhị phân (Binary Search Tree - BST) là một dạng cây nhị phân đặc biệt được sử dụng để tổ chức dữ liệu có thứ tự để thực hiện các phép tìm kiếm hiệu quả. Trong cây tìm kiếm nhị phân, mỗi nút có tối đa hai nút con: một nút con bên trái và một nút con bên phải.

Các thuộc tính chính của cây tìm kiếm nhị phân:

* **Thuộc tính tìm kiếm**: Đối với mỗi nút trong cây, tất cả các nút ở cây con bên trái của nút đó có giá trị nhỏ hơn nút đó và tất cả các nút ở cây con bên phải có giá trị lớn hơn nút đó.
* **Thuộc tính nhị phân**: Mỗi nút có tối đa hai nút con, một bên trái và một bên phải.
  1. Đồ thị.

Trong lĩnh vực tin học, đồ thị là một cấu trúc dữ liệu biểu diễn mối quan hệ giữa các đối tượng. Đối tượng này có thể là các điểm dữ liệu (đỉnh) và các mối quan hệ giữa chúng (cạnh). Mỗi cạnh thường có một hướng (được gọi là đồ thị có hướng) hoặc không có hướng (được gọi là đồ thị vô hướng).

Các ứng dụng của đồ thị rất đa dạng, từ mô hình hóa mạng lưới xã hội, đường đi trong mạng lưới, quy hoạch đồ thị (graph algorithms) đến biểu diễn cấu trúc dữ liệu trong lập trình và nhiều lĩnh vực khác. Đồ thị cung cấp một cách hiệu quả để mô hình hóa và giải quyết các vấn đề phức tạp trong nhiều lĩnh vực khác nhau.

* 1. Đỉnh (Vertex).

Đỉnh là một thành phần cơ bản của đồ thị, thường được biểu diễn bằng một điểm. Mỗi đỉnh đại diện cho một đối tượng hoặc một thực thể trong bài toán cụ thể. Đỉnh thường chứa thông tin về đối tượng mà nó đại diện.

Trong một đồ thị có hướng, đỉnh cũng có thể được gọi là node. Mỗi đỉnh trong đồ thị có thể kết nối với các đỉnh khác thông qua các cạnh, biểu thị mối quan hệ giữa các đối tượng tương ứng.

* 1. Cạnh (Edge).

Cạnh là một đoạn đường hoặc liên kết giữa hai đỉnh (hoặc node) trong đồ thị. Cạnh biểu diễn mối quan hệ giữa các đối tượng mà các đỉnh đại diện. Mỗi cạnh thường có thể có một hướng (được gọi là cạnh có hướng) hoặc không có hướng (được gọi là cạnh vô hướng).

Cạnh thường có thể mang theo thông tin bổ sung, như trọng số hoặc chiều dài, phụ thuộc vào loại đồ thị và bài toán cụ thể. Ví dụ, trong một đồ thị đường đi, trọng số có thể biểu thị chi phí hoặc khoảng cách giữa các đỉnh liên kết bởi cạnh.

* 1. Đường đi

Dãy các node hoặc cạnh trong cấu trúc liên kết mà có thể được kết nối bằng cách đi qua các node khác.

* 1. Thành phần liên thông.

Thành phần liên thông là một khối đồ thị mà ở đó mọi cặp đỉnh (node) đều có đường đi giữa chúng. Nói cách khác, trong một thành phần liên thông, bạn có thể đi từ một đỉnh bất kỳ đến bất kỳ đỉnh nào khác trong thành phần đó bằng cách theo các cạnh của đồ thị.

* 1. Đường găng:

"Critical path" (Đường găng) là chuỗi các công việc trong một dự án mà nếu bất kỳ công việc nào trên đường này bị trì hoãn, thì toàn bộ dự án sẽ bị trì hoãn.

* 1. Thuật toán đường găng.

Thuật toán đường găng thường được sử dụng để xác định các công việc quan trọng nhất và đặt ra một lịch trình mà nếu được theo đúng, dự án sẽ được hoàn thành đúng hạn.

1. **Bài toán đặt ra.**
   1. Bài toán

Bạn là quản lý một dự án phần mềm lớn có nhiều thành viên tham gia. Dự án của bạn bao gồm nhiều công đoạn khác nhau, từ việc lập kế hoạch, phát triển, kiểm thử đến triển khai. Mỗi thành viên trong nhóm đều có các nhiệm vụ và deadline riêng, và có sự phụ thuộc giữa các công việc. Bạn cần một cách hiệu quả để theo dõi tiến độ, quản lý tài nguyên, và giữ cho mọi người đồng bộ trong quá trình làm việc.

* 1. Vấn đề
* **Đồng bộ hóa thông tin:** Thông tin về tiến độ công việc, tài nguyên sử dụng, và các thay đổi trong dự án cần phải được đồng bộ hóa một cách nhanh chóng và chính xác.
* **Quản lý nguồn lực:** Để đảm bảo rằng các thành viên có đủ tài nguyên để hoàn thành công việc của họ, cần một phương tiện để theo dõi và quản lý nguồn lực dự án.
* **Theo dõi tiến độ:** Cần một công cụ mạnh mẽ để theo dõi tiến độ của từng công việc và phân đoạn của dự án, giúp phát hiện và giải quyết vấn đề kịp thời.
* **Giao tiếp hiệu quả:** Việc giao tiếp giữa các thành viên và các bên liên quan là quan trọng. Một phần mềm quản lý dự án cần hỗ trợ giao tiếp liền mạch và lưu trữ thông tin quan trọng.
  1. Tính cần thiết.

Cần thiết của phần mềm quản lý dự án: Một phần mềm quản lý dự án hiệu quả sẽ giúp giải quyết những thách thức trên, bằng cách cung cấp các tính năng như theo dõi tiến độ, quản lý nguồn lực, giao tiếp tự nhiên, và đồng bộ hóa thông tin. Điều này sẽ giúp tăng cường hiệu suất làm việc, giảm thiểu rủi ro, và đảm bảo rằng dự án được triển khai một cách thành công.

Trong một dự án lớn, việc có quá nhiều task và các nhân lực khác nhau sẽ khiến cho việc phân bổ tương đối khó khăn, vì vậy cần có giải pháp để tự động hóa công tác này một cách tự động và nhanh chóng.

1. **Các cấu trúc dữ liệu.** 
   1. ArrayList
      1. Giới thiệu

Được cài đặt trong lớp `ArrayList.java` ở trong gói Model. mô phỏng một phiên bản đơn giản của danh sách mảng có thể thay đổi kích thước trong Java. Các thuộc tính, phương thức sẽ được ghi chú bằng javadoc tương ứng trong file java gửi kèm cuối khóa học.

Lớp này sử dụng generics để hỗ trợ việc lưu trữ các kiểu dữ liệu khác nhau. Các phương thức như add, insert, remove, và toArray được triển khai để quản lý kích thước của mảng và đảm bảo rằng danh sách có thể thay đổi kích thước một cách linh hoạt. Đồng thời, lớp cũng cung cấp các phương thức như getClassName để truy cập thông tin về loại dữ liệu được lưu trữ trong danh sách.

* + 1. Mục đích sử dụng

Để lưu trữ các dữ liệu, các dự án, task,… theo danh sách để dễ dàng truy cập trong việc các thao tác với dự án. Cung cấp vùng chứa để dễ dàng thao tác với các đối tượng.

* 1. Đồ thị
     1. Giới thiệu

Cấu trúc dữ liệu này không được cài đặt trực tiếp dưới dạng một lớp, mà được cài đặt trong lớp Task.java theo dạng danh sách kề và được lưu trữ dưới dạng HashMap.

* + 1. Mục đích sử dụng

Các Task sẽ có các rằng buộc với nhau (Task trước phải hoàn thiện thì task sau mới được thực hiện. Nên việc biểu thị các mối liên hệ giữa các Task, giống như biểu thị một đồ thị. Việc xác định thứ tự hoàn thành công việc sẽ giúp công việc trở nên nhanh dễ dàng phân công và theo dõi tiến độ hơn.

* 1. Hash Map
     1. Giới thiệu

HashMap được cài đặt trong lớp HashMap.java ở trong gói Model. Mô phỏng cách hoạt động của bảng băm. Các thuộc tính và phương thức được ghi chú ở trong javadoc tương ứng trong file source code nộp cuối dự án.

* + 1. Mục đích sử dụng.

Ưu điểm của HashMap lớn nhất là truy vấn, tuy nhiên trong dự án này mục đích sử dụng chủ yếu để implement đồ thị.

* 1. Binary Searching Tree
     1. Giới thiệu

Được cài đặt trong ớp BinarySearchingTree.java ở trong gói Model. Mô phỏng lại cách hoạt động của một cây tìm kiếm nhị phân. Điểm đặc biệt của cây này là cân bằng chiều sâu của các node mỗi khi có sự thay đổi trên cây. Khóa được cài đặt trong dự án là Title (Tiêu đề) và khó tránh khỏi trường hợp có các tiêu đề trùng lặp. Chính vì thế, tại các node giá trị sẽ là một ArrayList chứa các task có cùng tiêu đề. Các thuộc tính và phương thức được ghi chú ở trong javadoc tương ứng trong file source code nộp cuối dự án.

* + 1. Mục đích sử dụng.

Như đã nói thì khối lượng phần tử là tương đối lớn, qua một cuộc khảo sát nhỏ và trải nghiệm, người dùng có thể đợi một khoảng nhỏ trong quá trình tạo nhưng sẽ khá khó chịu nếu thời gian tìm kiếm quá lâu. Chính vì thế, các dữ liệu được lưu trữ sẽ được add vào cây với Key được cài đặt là Title (Đối tượng đại diện cho Tiêu đề). Điều này sẽ khiến cho thời gian thêm dự án/ thêm task trở nên lâu hơn đôi chút, vì ngoài việc tạo thì còn phải tìm kiếm vị trí phù hợp. Tuy nhiên thì thời gian truy vấn trở nên nhanh hơn khi thuật toán tìm kiếm trên cây có độ phức tạp thời gian không quá lâu.

1. **Thuật toán sử dụng.**
   1. Tìm kiếm tuyến tính.

Tìm kiếm tuần tự tại các danh sách chưa được sắp xếp. Dễ dàng truy vấn và cài đặt. Ưu điểm là độ chính xác khá cao, và luôn trả về phần tử đầu tiên. Một số đối tượng đặc biệt không được kế thừa lớp Comparable sẽ không gặp vấn đề trong việc so sánh.

* 1. Tìm kiếm nhị phân

Là thuật toán tìm kiếm ưu việt hơn so với thuật toán tìm kiếm tuyến tính về mặt thời gian. Khi tìm kiếm sẽ loại bỏ một nửa các đối tượng mà hoàn toàn không thỏa mãn. Thuật toán sẽ không trả về phần tử đầu tiên mà trả về phần tử được tìm thấy đầu tiên theo quy tắc chia đôi. Nhược điểm lớn nhất của thuật toán này là yêu cầu phải sắp xếp các phần tử trong danh sách.

1. **Các đối tượng**
   1. Title

Đối tượng đại diện cho tiêu đề, chứa thông tin tiêu đề và mô tả. Trong việc hiển thị, đôi khi chỉ cần hiển thị tiêu đề không cần thiết việc hiển thị mô tả, tuy nhiên mô tả sẽ cần trong trường hợp cần diễn giải một cái gì đó phức tạp.

* 1. Repository

Hiện tại dự án đang hướng tới việc sử dụng cho các lập trình viên và việc sử dụng Repository của Github để lưu trữ code là một điều tốt nhất. Dễ dàng có thể theo dõi, làm việc trực tiếp với nhau mà không gây xung đột.

* 1. Label

Đối tượng là nhãn của các đối tượng khác. Mỗi công việc sẽ có nhãn khác nhau, việc truy vấn theo nhãn đôi khi sẽ dễ dàng hơn sẽ biết được chuyên môn của mình có phù hợp với nhãn của task không? Hoặc khi muốn tìm xem tiến độ của các task có nhãn cụ thể.

* 1. People

Đối tượng là đại diện cho một các nhân, có trách nhiệm lưu trữ thông tin, quản lý chuyên môn của người trong tổ chức. Hiện tại do là Desktop app chạy trên local nên chưa có nhiều thành viên khác nên thông tin chỉ có Tên và ID khi phức tạp hơn có thể sẽ cần thêm các thông tin khác để chắc chắn thành viên là không giống nhau.

* 1. Major

Đối tượng lưu trữ chuyên môn. Hiện tại đối tượng đang được fix cứng cho một số các vai trò và vị trí trong một dự án lập trình phần mềm. Thay đổi giá trị các thuộc tính để có được các giá trị về chuyên môn mong muốn.

* 1. Date

Đối tượng ngày tháng, biểu thị các thông tin về ngày tháng một cách khác. Việc sử dụng lớp Date tự cài đặt sẽ có điểm lợi hơn trong việc phát triển dự án. Và dễ dàng kiểm soát các thuộc tính của đối tượng.

* 1. Task

Đối tượng công việc, chứa các thuộc tính nhãn, tiêu đề, người thực hiện, ngày bắt đầu, kết thúc, mức độ quan trọng, rằng buộc,… các thông tin để hiển thị và giúp cho người sử dụng biết được tình trạng và tiến độ của công việc.

* 1. Project

Đối tượng chứa một danh sách các Task, thực hiện các thao tác với các task như gán nhãn, thay đổi trạng thái, phân công,… Quản lý các Task thông qua đối tượng Project để đồng bộ các Task trong một project với nhau một cách dễ dàng hơn.

* 1. ProjectManager

Được cài đặt theo mẫu thiết kế Singleton để đảm bảo rằng khi chạy ứng dụng, chỉ có duy nhất một Manager được tạo và những đối tượng khác đều chỏ đến đúng một manager.

1. **Lộ trình phát triển sau môn học.**

Hiện tại do có nhiều các môn học khác cũng cần hoàn thiện dự án, yêu cầu kỹ thuật và chuyên môn chưa thể đáp ứng được các kế hoạch nên nhóm đã hoàn thiện những vấn đề cơ bản nhất của dự án, còn đây là các vấn đề sẽ phát triển sau khóa học.

* 1. Sử dụng không gian mạng

Hiện tại dự án chỉ hình thành dưới dạng destop app, tức là dữ liệu chỉ được lưu trong máy tính cá nhân của một người dùng. Đồng nghĩa với việc ứng dụng sẽ phù hợp với những cá nhân chịu trách nhiệm quản lý, dễ dàng theo dõi công việc và điều phối. Điều này sẽ không thực tế vì khi làm việc việc cập nhật tiến độ sẽ do những người trực tiếp thực hiện cập nhật. Khi này trách nhiệm của người quản lý sẽ trở nên nặng nề và phức tạp hơn. Chính vì thế khi sử dụng không gian mạng sẽ cho phép người dùng thoải mái truy cập dưới các tài khoản như một số các dự án khác.

* 1. Đa nền tảng

Các công việc có thể dễ dàng xem trên máy tính, tuy nhiên trong các trường hợp khác như khi đi nghỉ hoặc đang trên đường, tham dự hội thảo,… đột nhiên có các công việc cần phân công người gấp, cần kiểm tra lại công việc sẽ dễ dàng hơn nếu sử dụng thiết bị điện thoại thay vì là phải sử dụng máy tính/ laptop để kiểm tra.

* 1. Trí tuệ nhân tạo trong việc tự động phân bổ nguồn lực, thời gian.

Hiện tại tính năng này đã có ở mức sơ khai nhất, tuy nhiên để mà thực sự gọi là trí tuệ nhân tạo cần một khoảng thời gian và sự phức tạp lớn hơn. Hiện tại tính năng tự động phân bổ được thực hiện như sau:

* Gán nhãn công việc cho các task
* Tạo các đối tượng People chứa các thông tin cơ bản và chứa thông tin về chuyên môn của người đó.
* Nhập danh sách người dùng vào project hiện có, Chương trình sẽ tự động phân bổ nguồn lực vào các task sao cho người thực hiện sẽ có chuyên môn chính xác, mỗi task sẽ có ít nhất một người đúng chuyên môn thực hiện.

Điều này khá là ổn nếu chỉ dừng ở mức cơ bản. Tuy nhiên có một điểm chưa được hợp lý ở ý tưởng này là các task có độ phức tạp khác nhau, mỗi người sẽ lại có chuyên môn khác nhau và quan trọng hơn hết, kinh nghiệm với các dự án khác cũng tương đối là khó xác định, chuyên môn của những người có cùng chuyên môn lại khá là khác nhau. Chưa có có nhiều rằng buộc khác trong thực tế phức tạp hơn và khó đưa về rạng toán học dẫn tới phân bổ không hợp lý. Chính vì thế tính năng sẽ chưa thể đề xuất một cách hoàn toán đúng những gì người dùng yêu cầu.

Về mặt thời gian thì việc xác định được độ khó theo thời gian sẽ khá phức tạp, thòi gian trong các dự án sẽ khó mà có thể cố định được mà còn phải dựa theo các yếu tố khác để xác định một cách chính xác.