BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN BINARY SEARCH VÀ LINEAR SEARCH

Giảng viên hướng dẫn: ThS. Bùi Thị Hồng Minh

Sinh viên thực hiện: Nguyễn Đức Hoàng Phi

Mã số sinh viên: 62131545

KHÁNH HÒA-2023

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN BINARY SEARCH VÀ LINEAR SEARCH

Giảng viên hướng dẫn: ThS. Bùi Thị Hồng Minh

Sinh viên thực hiện: Nguyễn Đức Hoàng Phi

Mã số sinh viên: 62131545

Khánh Hòa, tháng 01/2023

TRƯỜNG ĐẠI HỌC NHA TRANG
Khoa: Công nghệ Thông tin

PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ BÁO CÁO THỰC TẬP CƠ SỞ

Tên đề tài: XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN BINARY SEARCH VÀ LINEAR SEARCH

Giảng viên hướng dẫn: ThS. Bùi Thị Hồng Minh

Sinh viên được hướng dẫn: Nguyễn Đức Hoàng Phi

MSSV: 62131545

Khóa: 62 **Ngành:** Công nghệ Thông tin

Lần	Ngày	Nội dung	Nhận xét của GVHD
1	7/12/2022	Nhận đề tài hướng dẫn và định hướng giải quyết vấn đề. Sinh viên trình bày kế hoạch thực hiện.	
2	14/12/2022	Sinh viên trình bày việc mô phỏng thuật toán chính dựa trên kiến thức đã được học ở môn kỹ thuật đồ họa và các kiến thức thu nhận được từ Internet để minh họa bài toán đa dạng nhất có thể.	
3	21/12/2022	Sinh viên hoàn thiện các thuật toán đã đề ra với dữ liệu đầu vào được chỉ định sẵn. Trình bày thuật toán với các trường hợp sai và chỉ ra được hướng khắc phục cho các trường hợp đó.	
4	2/01/2023	Sinh viên nộp bản thảo của báo cáo thực tập lần thứ 1 và tiến hành chỉnh sửa.	

Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KL):

Khánh Hòa, ngày 02 tháng 01 năm 2021

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để có thể hoàn thành đợt thực tập lần này, em xin chân thành cảm ơn đến quý thầy cô khoa Công nghệ Thông tin đã tạo điều kiện hỗ trợ và giúp đỡ em trong quá trình học tập và nghiên cứu đề tài này.

Qua đây, em xin chân thành cảm ơn cô Bùi Thị Hồng Minh, người đã trực tiếp quan tâm và hướng dẫn em hoàn thành tốt đợt thực tập trong thời gian qua.

Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

MỤC LỤC

TÓM TẮT

Trong ngành khoa học máy tính, một giải thuật tìm kiếm là một thuật toán lấy đầu vào là một bài toán và trả về kết quả là một lời giải cho bài toán đó, thường là sau khi cân nhắc giữa một loạt các lời giải có thể. Hầu hết các thuật toán được nghiên cứu bởi các nhà khoa học máy tính để giải quyết các bài toán đều là các thuật toán tìm kiếm. Tập

hợp tất cả các lời giải có thể đối với một bài toán được gọi là không gian tìm kiếm. Thuật toán thử sai (brute-force search) hay các thuật toán tìm kiếm "sơ đẳng" không có thông tin sử dụng phương pháp đơn giản nhất và trực quan nhất. Trong khi đó, các thuật toán tìm kiếm có thông tin sử dụng heuristics để áp dụng các tri thức về cấu trúc của không gian tìm kiếm nhằm giảm thời gian cần thiết cho việc tìm kiếm.

Mô phỏng về thuật toán tìm kiếm nhị phân(Binary Search) và tìm kiếm tuyến tính(Linear Search) thực sự là một đề tài khá thú vị và hấp dẫn nhưng cũng gặp đôi chút khó khăn bởi vì tính trừu tượng của giải thuật, khó hình dung vấn đề. Do đó chúng ta luôn muốn có những chương trình mô phỏng trực quan để có thể tiếp thu một cách dễ dàng hơn.

Toàn bộ mã nguồn của báo cáo được tải lên theo địa chỉ:

https://github.com/phiahihi/search_algorithm

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Định nghĩa và khái niệm tìm kiếm

Tìm kiếm là một đòi hỏi rất thường xuyên trong các ứng dụng tin học. Bài toán tìm kiếm có thể phát biểu như sau:

- Cho một dãy gồm n bản ghi r[1...n]. Mỗi bản ghi r[i] ($1 \leq i \leq n$) tương ứng với một khóa k[i]. Hãy tìm bản ghi có giá trị khóa bằng X cho trước. X được gọi là khóa tìm kiếm hay đối trị tìm kiếm (argument).
- Công việc tìm kiếm sẽ hoàn thành nếu như có một trong hai tình huống sau xảy ra:
    + Tìm được bản ghi có khóa tương ứng bằng X, lúc đó phép tìm kiếm thành công.
    + Không tìm được bản ghi nào có khóa tìm kiếm bằng X cả, phép tìm kiếm thất bại.

1.2. Sử dụng thuật toán tìm kiếm nhị phân để giải quyết các bài toán tìm kiếm

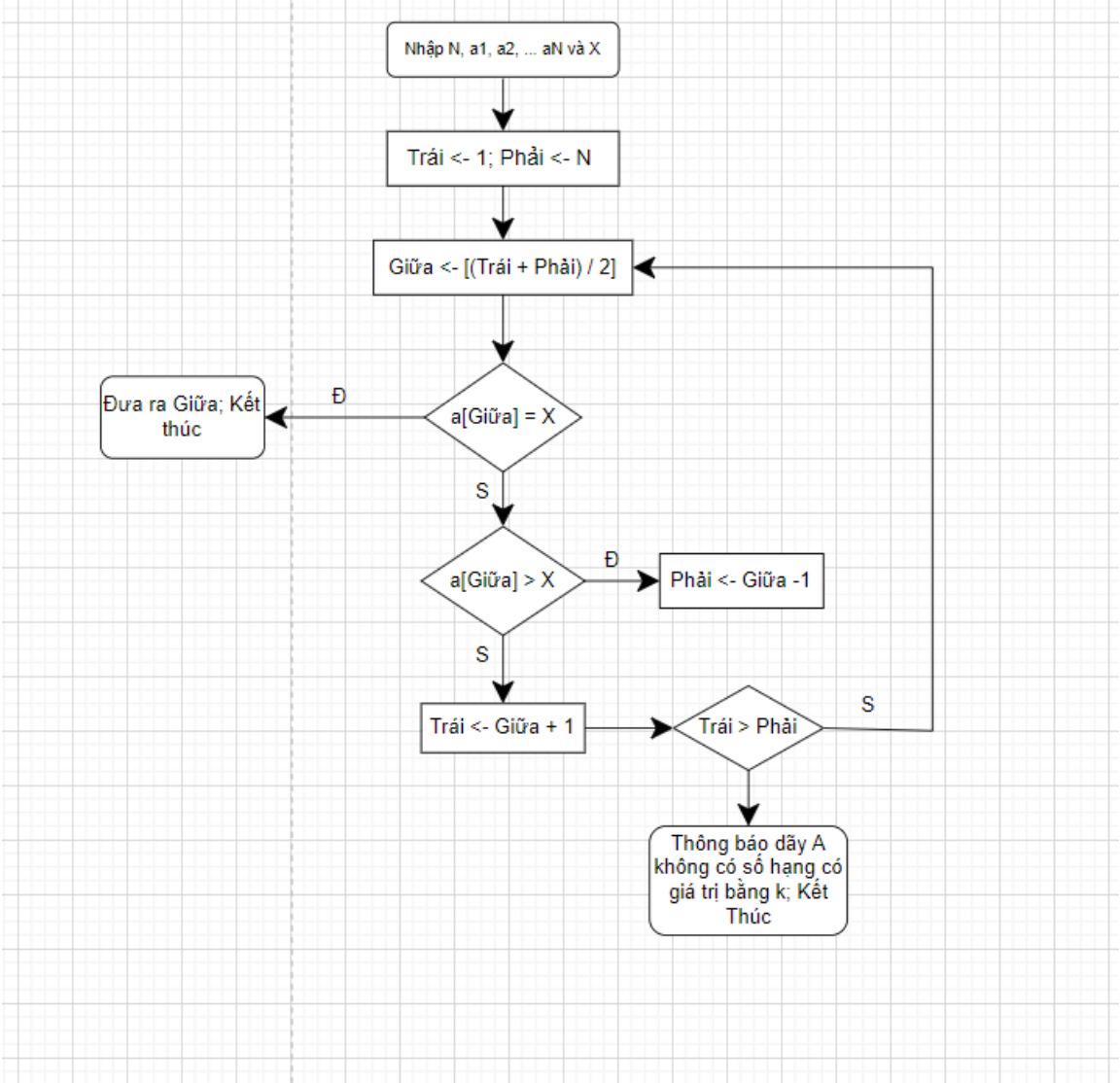
- Phép có thể áp dụng trên dãy khóa đã có thứ tự: $k[1] \leq k[2] \leq ... k[n]$. Giả sử ta cần tìm trong đoạn k[left...right] với khóa tìm kiếm là X, trước hết ta xét khóa nằm giữa gãy k[mid] với mid = (left + right) div 2;

    + Nếu k[mid] < X thì có nghĩa là đoạn từ k[left] tới k[right] chỉ chứa toàn khóa < X, ta tiến hành tìm kiếm tiếp với đoạn từ k[mid + 1] tới k[right].

    + Nếu k[mid] > X thì có nghĩa là đoạn từ k[mid] tới k[right] chỉ chứa toàn khóa > X, ta tiến hành tìm kiếm tiếp với đoạn từ k[left] tới k[mid -1].

    + Nếu k[mid] = X thì việc tìm kiếm thành công (kết thúc quá trình tìm kiếm)

- Quá trình tìm kiếm sẽ thất bại nếu đến một bước nào đó, đoạn tìm kiếm là rỗng (left > right).
- Sơ đồ các bước thực hiện thuật toán tìm kiếm nhị phân (Binary search):



{Tìm kiếm nhị phân trên dãy khóa k[1] ≤ k[2] ≤ ... k[n]; hàm này thử tìm xem trong dãy có khóa nào = X không, nếu thấy nó trả về chỉ số của khóa ấy, nếu không thấy nó trả về -1}

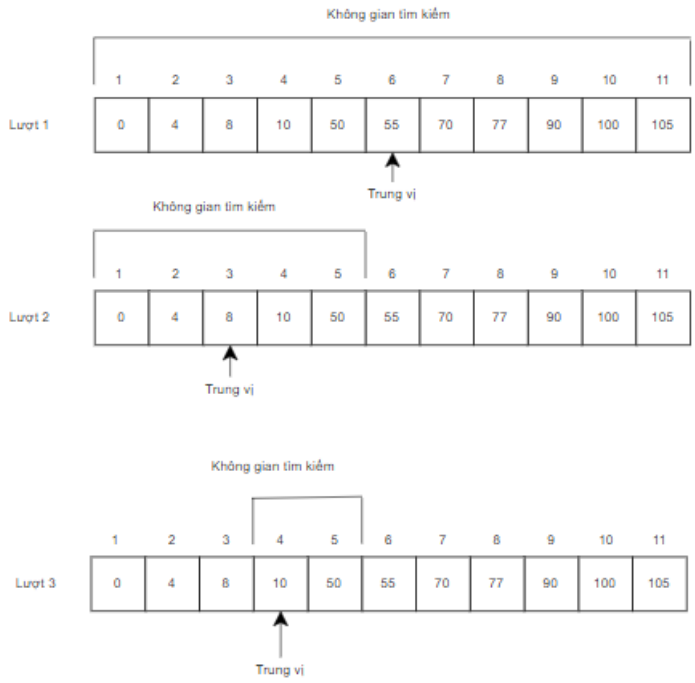
```
int _binarySearchHelper(List<int> list, int target, int left, int right) {
  while (left <= right) {
    final middle = (left + right) ~/ 2;
    final potentialMatch = list[middle];
    if (target == potentialMatch) {
      return middle;
    } else if (target < potentialMatch) {
      right = middle - 1;
    } else {
      left = middle + 1;
    }
  }
  return -1;
}
```

Người ta đã chứng minh được độ phực tạp của thụât toán tìm kiếm nhị phân trong trường hợp tốt nhất là O(1), trong trường hợp xấu nhất là O(log2n) và trung bình cũng là O(log2n). Tuy nhiên, ta không nên quên rằng trước khi sử dụng tìm kiếm nhị phân, dãy khoá phải được sắp xếp rồi, tức là thời gian chi phí cho việc sắp xếp cũng phải tính đến.

Trường hợp	Số lần so sánh	Giải thích
Tốt nhất	1	Phần tử giữa mảng có giá trị X
Xấu nhất	Log2n	Không có X trong mảng
Trung bình	Log2(n/2)	Giả sử xác suất các phần tử trong mảng nhận giá trị là như nhau

Ví dụ:

Cho A = [0, 4, 8, 10, 50, 55, 70, 77, 90, 100, 105] và x = 10, thuật toán sẽ diễn ra Như hình dưới:



Ở lượt tìm đầu tiên, không gian tìm kiếm là tập hợp S=1,…,11 gồm tất cả các chỉ số của mảng. Bắt đầu với việc chọn phần tử trung vị của không gian tìm kiếm hiện tại (chính là 6), ta nhận xét A[6]=55 > 10 =x. Do theo đề bài mảng A được sắp xếp tăng dần, ta biết được tất cả các phần tử có chỉ số 6,…,11 đều lớn hơn giá trị cần tìm x. Do đó, chúng chắc chắn không thể là kết quả, khi đó không gian tìm kiếm có thể thu hẹp lại S=1,…,5, tức giảm đi một nửa.

Tương tự, ở lượt tìm thứ hai, ta xét phần tử trung vị của không gian tìm kiếm hiện tại (chính là 3), nhận thấy A[3] = 8 < 10 = x. Cũng do mảng A được sắp xếp tăng dần, ta biết được tất cả các phần tử có vị trí từ 1,…,3 đều bé hơn giá trị cần tìm x. Do đó, chúng chắc chắn không thể là kết quả, khi đó không gian tìm kiếm sẽ lại giảm đi một nửa S=4,…,5.

Ở lượt tìm cuối cùng, ta cũng xét phần tử trung vị của không gian tìm kiếm hiện tại ở vị trí 4 (ở đây số lượng phần tử của không gian tìm kiếm là chẵn, do đó có hai phần tử trung vị, ta có thể chọn một trong hai đều được, ở ví dụ này ta chọn phần tử trung vị đầu tiên), Nhận thấy A[4] = 10 = x, ta kết luận 4 chính là vị trí của phần tử cần tìm và dừng thuật toán.

1.3. Sử dụng thuật toán tìm kiếm tuyến tính để giải quyết các bài toán tìm kiếm

- So sánh x lần lượt với phần tử thứ 1, thứ 2,…của danh sách cho đến khi gặp được phần tử cần tìm. Nếu tìm hết mảng mà không thấy thì xuất thông báo không tìm thấy.

Các bước của thuật toán:

Bước 1: Gán i = 0; //bắt đầu từ phần tử đầu tiên của mảng

Bước 2: So sánh a[i] với x, 2 khả năng có thể xảy ra:

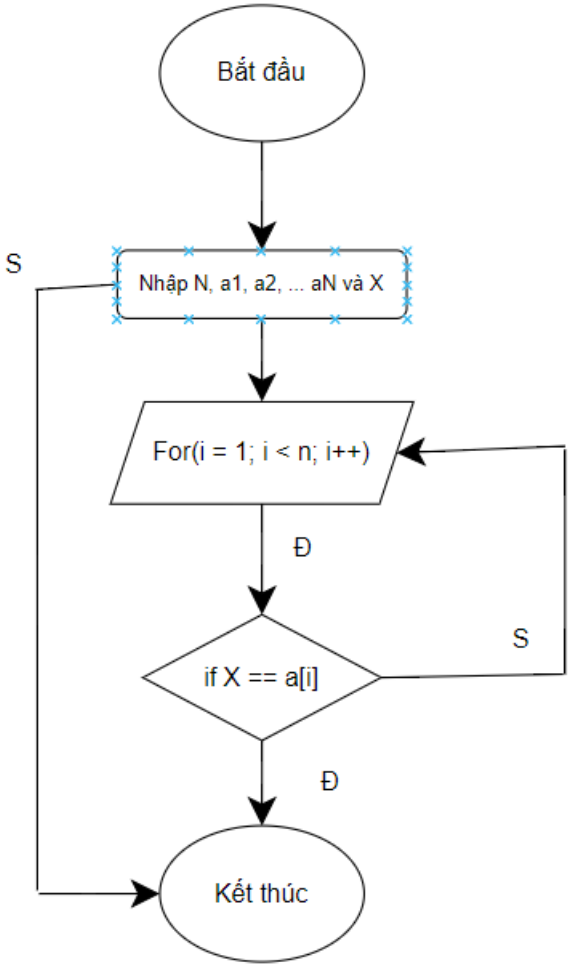
- a[i] == X: Tìm thấy -> Xuất kết quả là i -> Dừng.
- a[i] != X: Chuyển qua Bước 3.

Bước 3: i = i + 1; //xét tiếp phần tử kế tiếp trong mảng

Nếu i == n : Hết mảng -> không tìm thấy -> Dừng.

Ngược lại: Lặp lại Bước 2.

- Sơ đồ các bước thực hiện thuật toán tìm kiếm tuyến tính(Linear search):



{Tìm kiếm tuyến tính trên dãy khóa k[1] ≤ k[2] ≤ ... k[n]; hàm này thử tìm xem trong dãy có khóa nào = X không, nếu thấy nó trả về chỉ số của khóa ấy, nếu không thấy nó trả về -1}

```
int linearSearch(List<int> list, int target) {
  for (var i = 0; i < list.length; i++) {
    if (list[i] == target) {
      return i;
    }
  }
  return -1;
}
```

Trong trường hợp tốt nhất độ phức tạp của thuật toán này là O(1), trường hợp xấu nhất là O(n), trung bình cũng là O(n).

Trường hợp	Số lần so sánh	Giải thích
Tốt nhất	1	Phần tử đầu tiên trong mảng có giá trị X
Xấu nhất	n	Không có X trong mảng
Trung bình	n / 2	Trường hợp X nằm giữa mảng

Ví dụ: Cho A = [0, 4, 8, 10, 50, 55, 70, 77, 90, 100, 105] và x = 10, thuật toán sẽ diễn ra như hình dưới:

Ở lượt tìm đầu tiên, không gian tìm kiếm là tập hợp S=1,…,11 gồm tất cả các chỉ số của mảng. Bắt đầu với việc chọn phần tử đầu tiên (chính là 1), ta nhận xét A[1] = 0 != 10 = X không phải là giá trị cần tìm, sau đó tăng phần tử lên 1. Ở lượt thứ 2 tiếp tục nhận xét A[2] = 4 != 10 = X không phải là giá trị cần tìm, sau đó tăng phần tử lên 1. Ở lượt thứ 3, tiếp tục nhận xét A[3] = 8 != 10 = X không phải là giá trị cần tìm, sau đó tăng phần tử lên 1. Ở lượt thứ 4, tiếp tục nhận xét A[4] = 10 = X, ta kết luận 4 chính là vị trí của phần tử cần tìm và dừng thuật toán.

1.4. Sự khác biệt chính giữa Tìm kiếm tuyến tính và Tìm kiếm nhị phân

- Tìm kiếm tuyến tính có tính lặp đi lặp lại và sử dụng cách tiếp cận tuần tự. Mặt khác, tìm kiếm nhị phân thực hiện phương pháp phân chia và chinh phục.
- Độ phức tạp thời gian của tìm kiếm tuyến tính là O (N) trong khi tìm kiếm nhị phân có O (log 2 N).
- Thời gian trường hợp tốt nhất trong tìm kiếm tuyến tính là cho phần tử đầu tiên, tức là O (1). Ngược lại, trong tìm kiếm nhị phân, nó dành cho phần tử ở giữa, tức là O (1).
- Trong tìm kiếm tuyến tính, trường hợp xấu nhất để tìm kiếm một phần tử là N số so sánh. Ngược lại, đó là log 2 N số so sánh cho tìm kiếm nhị phân.
- Tìm kiếm tuyến tính có thể được thực hiện trong một mảng cũng như trong danh sách được liên kết trong khi tìm kiếm nhị phân không thể được thực hiện trực tiếp trên danh sách được liên kết.
- Như chúng ta biết Tìm kiếm nhị phân yêu cầu mảng được sắp xếp là lý do nó yêu cầu xử lý để chèn vào vị trí thích hợp của nó để duy trì danh sách được sắp xếp. Ngược lại, tìm kiếm tuyến tính không yêu cầu các yếu tố được sắp xếp, vì vậy các yếu tố dễ dàng được chèn vào cuối danh sách.
- Tìm kiếm tuyến tính rất dễ sử dụng và không cần bất kỳ yếu tố nào. Mặt khác, thuật toán tìm kiếm nhị phân tuy nhiên rất khó và các yếu tố nhất thiết phải được sắp xếp theo thứ tự.

1.5. Visual Studio Code

Visual Studio Code (https://code.visualstudio.com/download) chính là ứng dụng cho phép biên tập, soạn thảo các đoạn code để hỗ trợ trong quá trình thực hiện xây dựng, thiết kế website một cách nhanh chóng. Visual Studio Code hay còn được viết tắt là VS Code. Trình soạn thảo này vận hành mượt mà trên các nền tảng như Windows, macOS, Linux. Hơn thế nữa, VS Code còn cho khả năng tương thích với những thiết bị máy tính có cấu hình tầm trung vẫn có thể sử dụng dễ dàng.

Các tính năng của Visual Studio Code:

- Đa dạng ngôn ngữ lập trình giúp người dùng thỏa sức sáng tạo và sử dụng như HTML, CSS, JavaScript, C++,…

- Ngôn ngữ, giao diện tối giản, thân thiện, giúp các lập trình viên dễ dàng định hình nội dung.
- Các tiện ích mở rộng rất đa dạng và phong phú.
- Tích hợp các tính năng quan trọng như tính năng bảo mật (Git), khả năng tăng tốc xử lý vòng lặp (Debug),…
- Đơn giản hóa việc tìm quản lý hết tất cả các Code có trên hệ thống.

1.6 Flutter

Flutter là nền tảng phát triển ứng dụng đa nền tảng cho iOS và Android do Google phát triển. Flutter sử dụng ngôn ngữ DART cũng do Google phát triển và flutter cũng đã được sử dụng để tạo ra các ứng dụng native cho Google.

Ưu điểm:

- Mạnh về hiệu ứng, hiệu suất ứng dụng rất cao.
- Giao tiếp gần như trực tiếp với hệ thống
- Ngôn ngữ kiểu tĩnh nhưng với cú pháp hiện đại (tương tự JS, Python, Java), compiler linh động khi dùng AOT (cho sản phẩm cuối) và JIT (cho quá trình phát triển với hot reload)
- Có thể chạy được giả lập mobile ngay trên web, tiện cho việc phát triển. Các bộ đo lường chỉ số hiệu suất được hỗ trợ sẵn giúp lập trình viên kiểm soát tốt hiệu suất của ứng dụng.
- Có thể dùng để xây dựng các nền tảng gắn vào ứng dụng native để tăng hiệu suất.

Nhược điểm:

- Hầu hết bộ render UI đã được viết lại, không liên quan đến UI sẵn có trong UI Framework native nên memory sẽ dùng khá nhiều. Bên cạnh đó, UI không đi cùng với OS mà được phát triển riêng.
- Thường xuyên bổ sung các kiến thức về ngôn ngữ DART: Có rất ít lập trình viên hiểu biết rõ về Dart. Có nhiều trường hợp sau khi học xong sẽ dính liền luôn với Dart trong mảng phát triển các ứng dụng trên mobile, chứ không thể linh động như Python, JS có thể qua lại giữa AI, back, ...
- Mô hình dữ liệu rất mới: Nếu đã quen với Redux thì phải sẽ mất kha khá thời gian để học hỏi thêm về mô hình dữ liệu của Flutter mặc dù nó không quá khó.

CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cài đặt Visual Studio Code và Framework Flutter

- Tải file cài đặt phần mềm Visual Studio Code theo đường dẫn trong mục 1.5. Sau đó mở file vừa tải, và tiến hành cài đặt. Visual Studio Code được tiến hành cài đặt theo các bước:

- Bước 1. Tải file cài đặt Visual Studio Code cho Windows.
- Bước 2. Sau khi tải xong, chạy file VSCodeUserSetup.exe.
- Bước 3. Nhấp vào Next để cài đặt. Tiếp theo đồng ý điều khoản sử dụng.
- Bước 4: Lựa chọn nơi cài đặt (Nên để mặc định) sau đó nhấn Next.
- Bước 5. Các bước tiếp theo tiếp tục nhấn Next cho tới khi hoàn tất. Trong quá trình này, mình khuyên các bạn nên tích chọn vào 2 chức năng: (1) Add "Open with Code" action to Windows Explorer file context menu và (2) Add "Open with Code" action to Windows Explorer directory context menu. Việc này giúp bạn có thể click chuột phải vào thư mục sẽ có lựa chọn mở bằng VS Code.
- Bước 6. Cài đặt hoàn tất, bạn có thể trải nghiệm.

- Cài đặt Flutter cho Windows:

  Đầu tiên, bạn truy cập vào flutter.dev. Sau đó chọn hệ điều hành mà bạn đang sử dụng. Mình sẽ chọn Windows.

- Yêu cầu hệ thống

  + Để cài đặt Flutter thì máy tính của bạn cần đáp ứng những yêu cầu sau đây:

  + Windows 7 SP1 hoặc mới hơn (64-bit).

  + Dung lượng trống 400MB (không bao gồm dung lượng cho phần IDE/tools).

  + Các công cụ: Flutter phụ thuộc vào các công cụ có sẵn trong máy tính của bạn:

  + PowerShell 5.0 hoặc mới hơn (cái này đã được tích hợp sẵn vào Windows 10).

+ Git cho Windows 2.x, với tùy chọn chạy câu lệnh git từ cửa sổ lệnh Windows Command Prompt. (Nếu Git đã được cài, hãy chắc chắn rằng bạn có thể chạy câu lệnh git từ cửa sổ lệnh Windows Command Prompt).

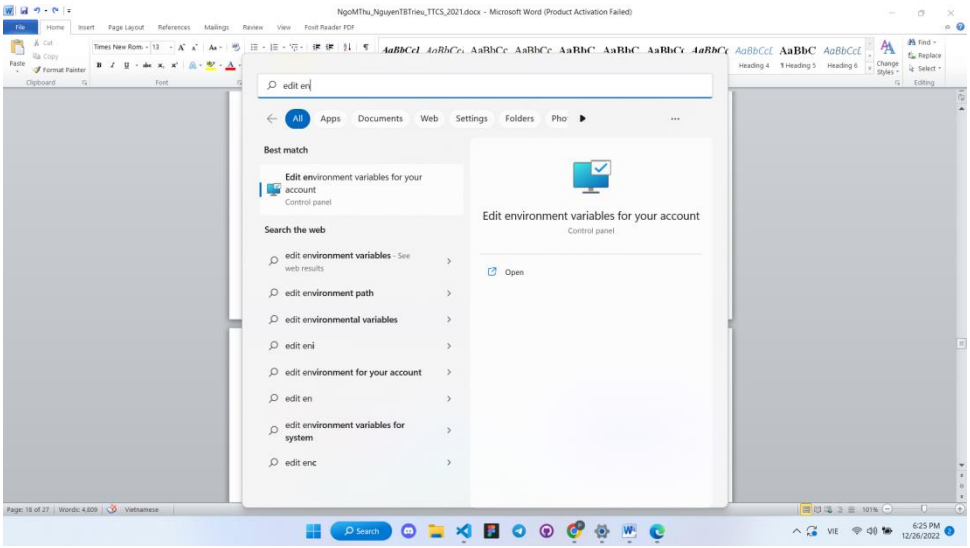
- Tiến hành cài đặt

+ Bạn duyệt tìm đến phần Get the Flutter SDK, bấm vào flutter_windows_vxxxx-stable.zip (với xxxx là phiên bản flutter, ví dụ flutter_windows_v1.9.1+hotfix.6-stable.zip) để tải flutter SDK về. Bạn có thể lưu ở đâu tùy thích sau khi giải nén mình có thể di chuyển nó sau.

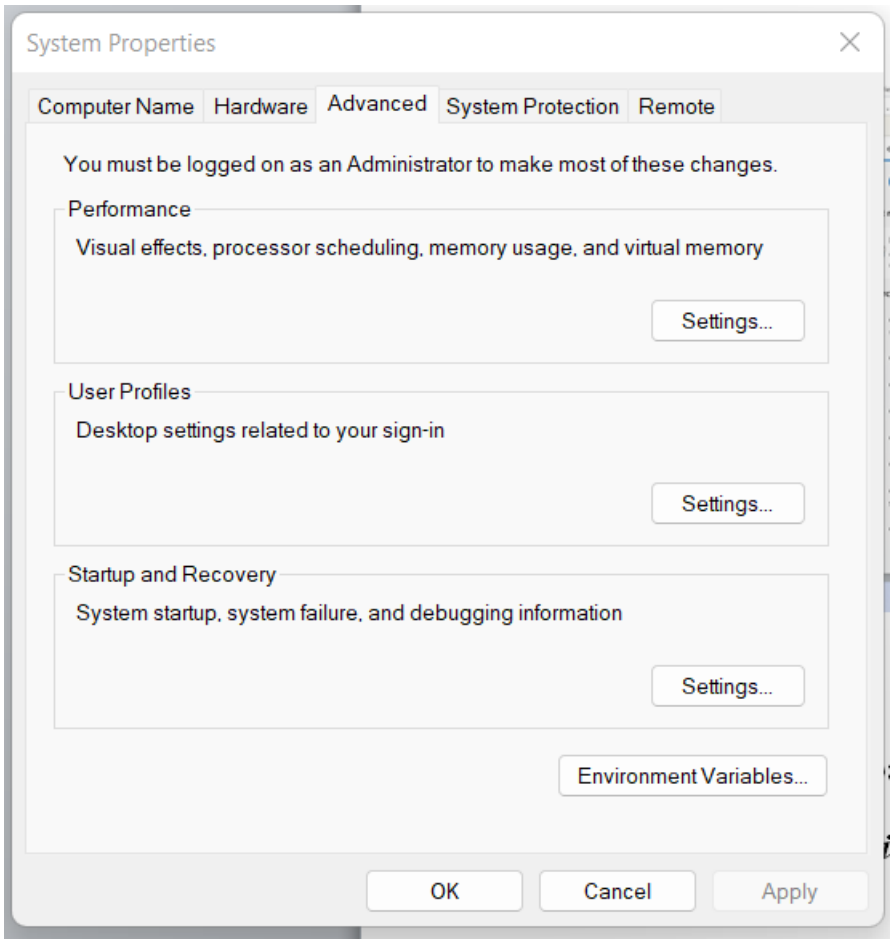
+ Sau khi tải về xong, các bạn giải nén file vừa tải về (tùy chọn Extract here) ta sẽ được một thư mục tên là flutter. Các bạn hãy để thư mục này vào một nơi nào đó mà bạn muốn (lưu ý không được đặt vào thư mục "C:\Program Files\" vì nó yêu cầu quyền riêng tư). Ví dụ mình sẽ làm y như trong docs của Google là tạo một thư mục mới tên là src đặt trong ổ đĩa "C:\" và copy thư mục flutter vào thư mục "C:\src\".

+ Bây giờ bạn đã sẵn sàng để chạy lệnh Flutter Console. Nhưng để có thể chạy lệnh flutter từ Command Prompt, bạn nên cập nhật đường dẫn. Các bước thực hiện như sau:

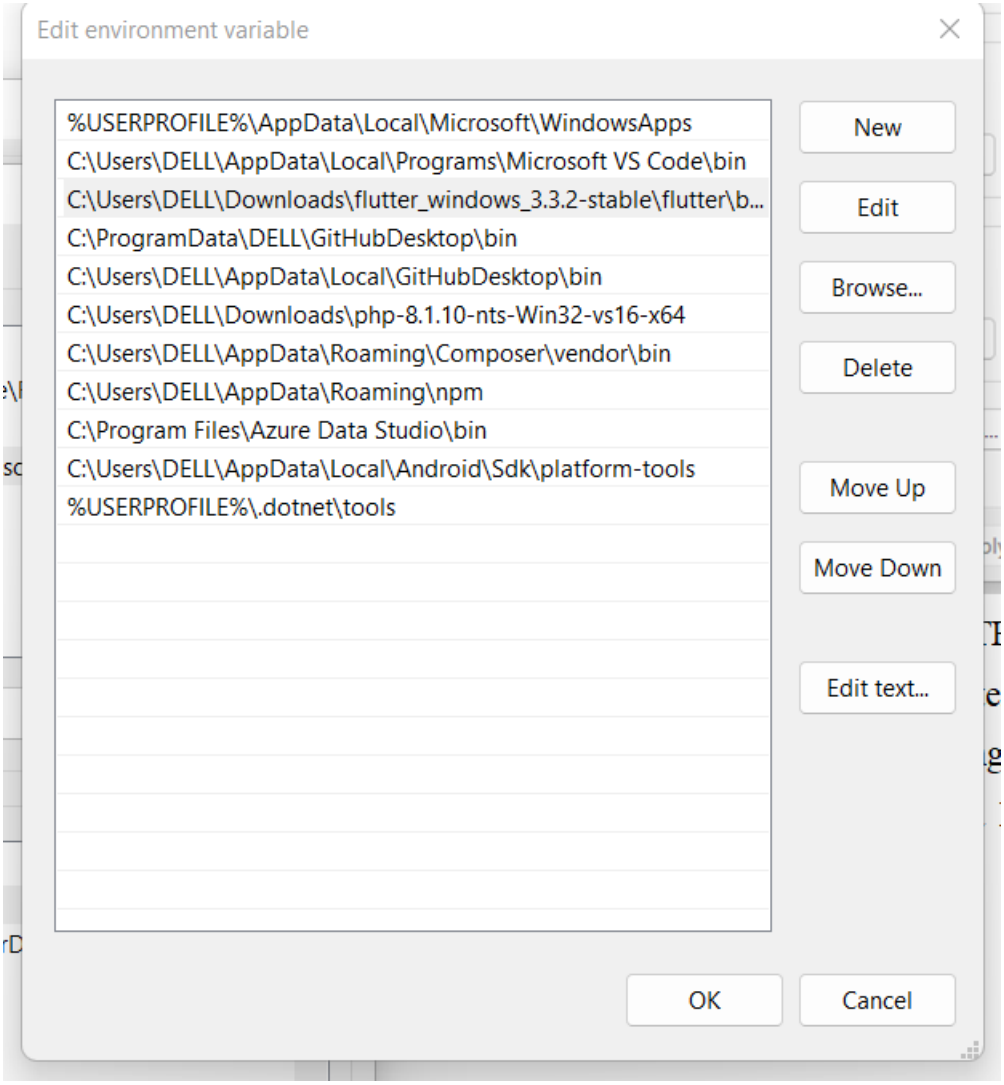
1. Search trên thanh công cụ Edit Enviroment Variable For Your Account



2. Chọn Enviroment Variables… trong cửa sổ System Properties.



3. Trong phần User variables, các bạn tìm Variable là PATH và nhấn Edit, nhấn New và thêm đường dẫn đến thư mục “flutter\bin”. Ví dụ của mình là “C:\src\flutter\bin”. Nếu như bạn không tìm thấy Variable PATH bạn nhấn New và đặt tên Variable là PATH và đường dẫn đến thư mục flutter\bin của bạn.



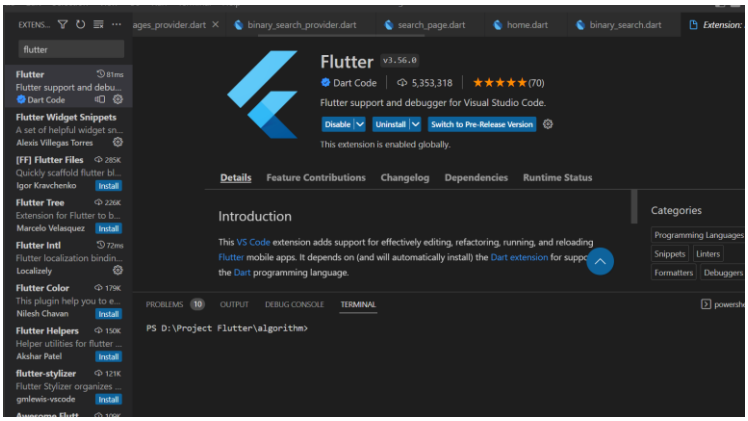
4. Nhấn OK để lưu tất cả các thay đổi lại.

- Cài đặt Plugins Flutter

1. Mở Visual Studio Code

2. Vào tab Extentions trên thanh sidebar bên ttrá.

3. Search “flutter”, các bạn nhấn Install plugin đầu tiên của kết quả tìm kiếm. Khác với Android Studio, khi cài plugin Flutter thì plugin Dart



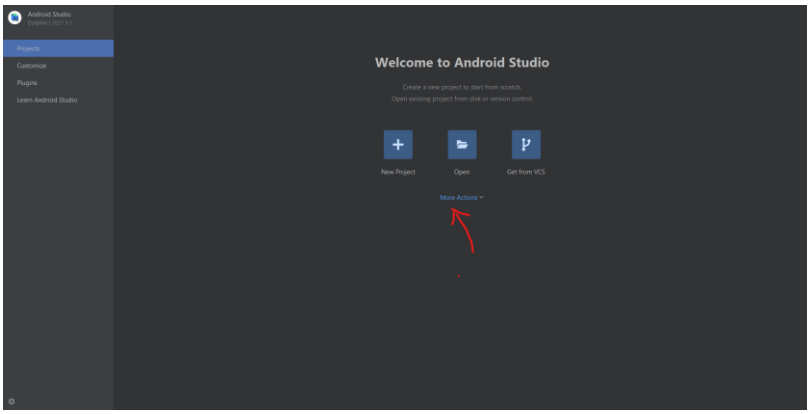
cũng sẽ được tự cài theo mà
không cần bạn xác nhận.

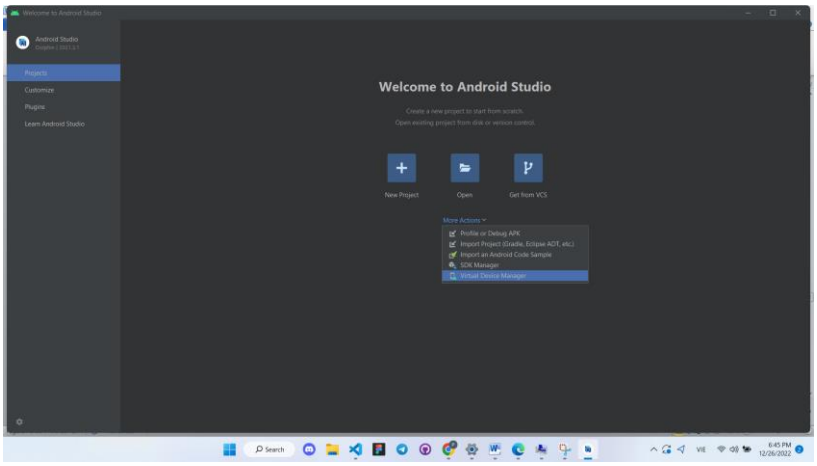
- Tạo máy ảo Android Studio

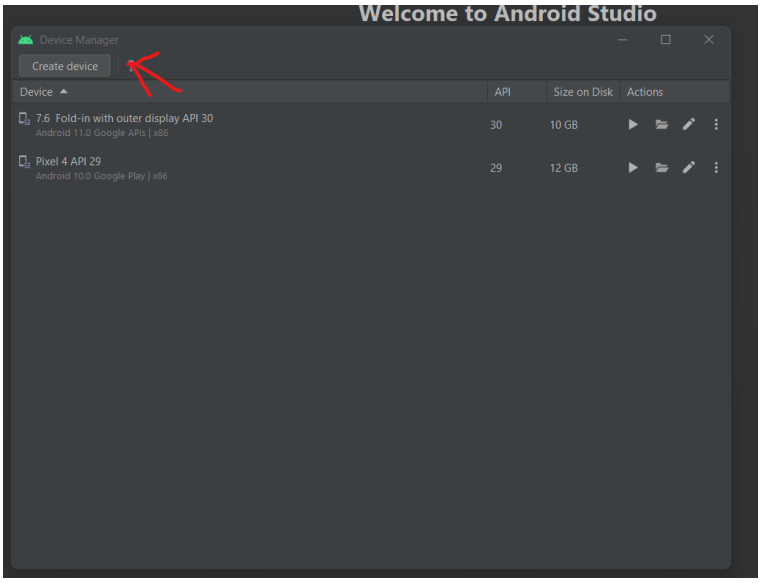
Muốn tạo máy ảo trước tiên bạn phải download Android Studio:
https://developer.android.com/studio

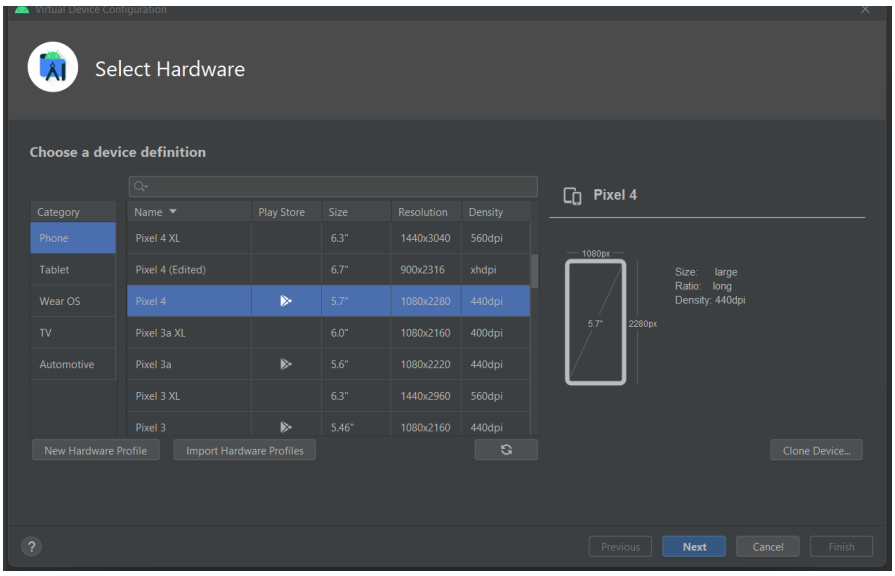
Để tạo một máy ảo Android Studio, các bạn làm theo các bước sau:

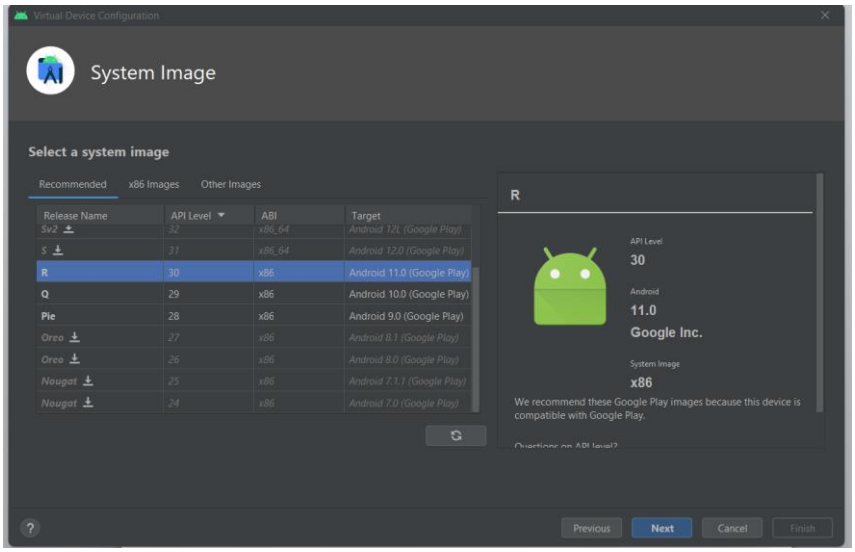
1. Khởi động Android Studio
2. Vào menu Configure và chọn AVD Manager
3. Chọn Create Virtual Device
4. Chọn một thiết bị và nhấn Next
5. Các bạn chọn một image x86 hoặc x86_64 đều được (khuyến khích bản mới nhất – nếu không có sẵn thì bạn nhấn Download và đợi một chút để nó tải image đó) sau đó nhấn Next
6. Trong phần Graphics, bạn chọn Hardware – GLES 2.0. Bạn có sửa các tùy chọn khác nếu muốn và nhấn Finish để tạo máy ảo.Nhấn nút ▶ để chạy máy ảo vừa tạo.

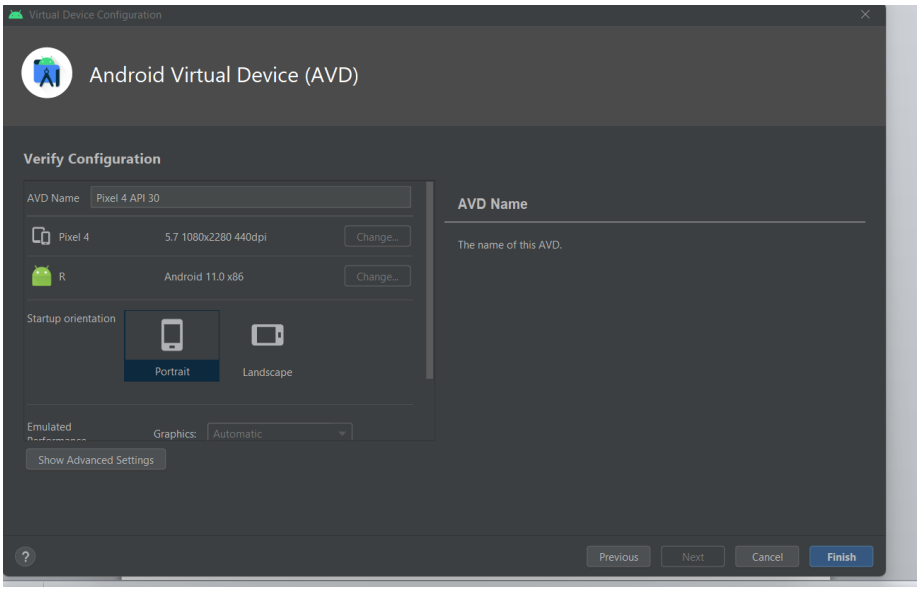


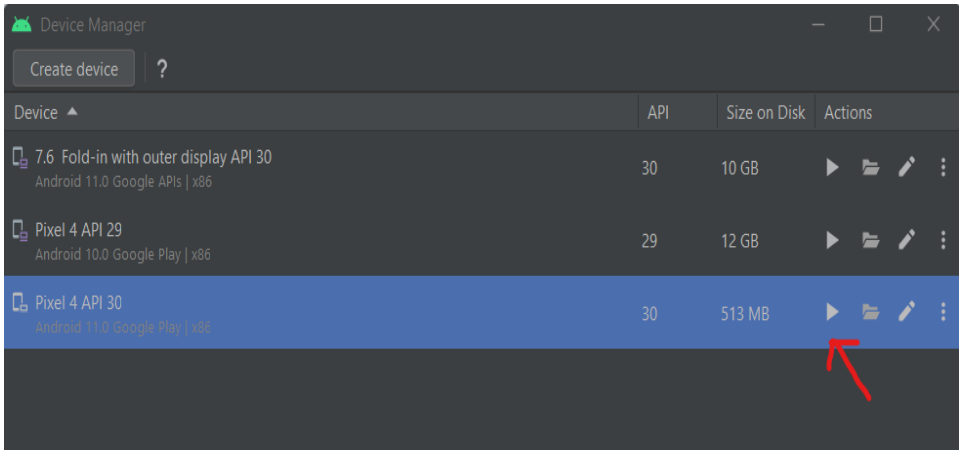

Welcome to Android Studio
New Project
Open
Get from VCS


Welcome to Android Studio
Device Manager
Create device
Device
API
Size on Disk
Actions
7.6 Fold-in with outer display API 30
Android 11.0 Google APIs | x86
30
10 GB
Pixel 4 API 29
Android 10.0 Google Play | x86
29
12 GB


Select Hardware
Choose a device definition
Category
Name
Play Store
Size
Resolution
Density
Phone
Pixel 4 XL
6.3"
1440x3040
560dpi
Tablet
Pixel 4 (Edited)
6.7"
900x2316
xhdpi
Wear OS
Pixel 4
5.7"
1080x2280
440dpi
TV
Pixel 3a XL
6.0"
1080x2160
400dpi
Automotive
Pixel 3a
5.6"
1080x2220
440dpi
Pixel 3 XL
6.3"
1440x2960
560dpi
Pixel 3
5.46"
1080x2160
440dpi
Pixel 4
1080px
2280px
Size: large
Ratio: long
Density: 440dpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish


Virtual Device Configuration
System Image
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
R
30
x86
Android 11.0 (Google Play)
Q
29
x86
Android 10.0 (Google Play)
Pie
28
x86
Android 9.0 (Google Play)
API Level
30
Android
11.0
Google Inc.
System Image
x86
We recommend these Google Play images because this device is compatible with Google Play.
Previous
Next
Cancel
Finish


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD Name
Pixel 4 API 30
Pixel 4
5.7 1080x2280 440dpi
Change...
R
Android 11.0 x86
Change...
Startup orientation
Portrait
Landscape
Graphics:
Automatic
Show Advanced Settings
AVD Name
The name of this AVD.
Previous
Next
Cancel
Finish


Device Manager
Create device
Device
API
Size on Disk
Actions
7.6 Fold-in with outer display API 30
Android 11.0 Google APIs | x86
30
10 GB
Pixel 4 API 29
Android 10.0 Google Play | x86
29
12 GB
Pixel 4 API 30
Android 11.0 Google Play | x86
30
513 MB

2.2. Cài đặt hàm cho ứng dụng

```
Future wait({double speed = 0.5}) {
  final milliseconds = lerpDouble(100, 2000, speed)!.toInt();
  return Future.delayed(Duration(milliseconds: milliseconds));
}
```

Hàm dùng chỉnh tốc độ khi di chuyển qua các giá trị tìm kiếm mặc định là 0.5 giây

```
void render() {
  notifyListeners();
}
```

Hàm dùng để rebuild lại UI khi dữ liệu có sự thay đổi

```
double _executionSpeed = 0.5;
double get executionSpeed => _executionSpeed;
set executionSpeed(double speed) {
  if (speed > 1.0) {
    _executionSpeed = 1;
    return;
  }
  if (speed < 0) {
    _executionSpeed = 0;
    return;
  }
  _executionSpeed = speed;
  render();
}
```

Hàm dùng để thiết lập tốc độ trong khoảng 0 – 1 giây

```dart
abstract class BaseProvider extends ChangeNotifier {
  double _executionSpeed = 0.5;
  double get executionSpeed => _executionSpeed;
  set executionSpeed(double speed) {
    if (speed > 1.0) {
      _executionSpeed = 1;
      return;
    }
    if (speed < 0) {
      _executionSpeed = 0;
      return;
    }
    _executionSpeed = speed;
    render();
  }

  void render() {
    notifyListeners();
  }

  Future pause() async {
    await wait(speed: executionSpeed);
  }
}
```

ChangeNotifier cho phép truy cập vào notifyListeners() và khi gọi notifyListeners() thì dữ liệu được thay đổi và màn hình được rebuild lại.

```dart
enum SearchState { open, discard, search, searched, found }

class SearchModel {
  SearchModel(this.value)
      : state = ValueNotifier(SearchState.open),
        color = Colors.black54,
        key = GlobalKey();

  final int value;
  ValueNotifier<SearchState> state;
  Color color;
  GlobalKey key;

  void reset() {
    state.value = SearchState.open;
    color = Colors.black54;
  }

  void potential() {
    state.value = SearchState.search;
    color = Colors.blue;
  }

  void discard() {
    state.value = SearchState.discard;
    color = Colors.red;
  }

  void found() {
    state.value = SearchState.found;
    color = Colors.green;
  }

  void searched() {
    state.value = SearchState.searched;
  }
}
```

Giá trị bao gồm 5 trạng thái open, discard, search, searched, found, và SearchModel giống như "khuôn" của giá trị và được mặc định có màu đen và trạng thái là open.

Hàm reset: được đặt trạng thái open và màu đen.

Hàm potential: được đặt trạng thái search và màu xanh.

Hàm discard: được đặt trạng thái discard và màu đỏ.

Hàm found: được đặt trạng thái found và có màu xanh lá

Hàm searched: được đặt trạng thái searched.

```dart
abstract class SearchProvider extends BaseProvider {
  final List<SearchModel> numbers = [
    SearchModel(0),
    SearchModel(1),
    SearchModel(4),
    SearchModel(11),
    SearchModel(19),
    SearchModel(22),
    SearchModel(34),
    SearchModel(35),
    SearchModel(38),
    SearchModel(39),
    SearchModel(44),
    SearchModel(46),
    SearchModel(47),
    SearchModel(49),
    SearchModel(57),
    SearchModel(62),
    SearchModel(69),
    SearchModel(74),
  ];

  bool _isSearching = false;
  int _position = -2;

  bool get isSearching => _isSearching;
  int get position => _position;
```

```dart
  void search({int value = 34}) {
    reset();
    _isSearching = true;
  }

  void reset() {
    _isSearching = false;
    _position = -2;
    for (var number in numbers) {
      number.reset();
    }
    notifyListeners();
  }

  void potentialNode(int index) {
    numbers[index].potential();
    notifyListeners();
  }

  void searchedNode(int index) {
    numbers[index].searched();
    notifyListeners();
  }

  void discardNode(int index) {
    numbers[index].discard();
    notifyListeners();
  }

  void discardNodes(int left, int right) {
    for (var index = left; index <= right; index++) {
      numbers[index].discard();
    }
    notifyListeners();
  }

  void foundNode(int index) {
    _isSearching = false;
    numbers[index].found();
    _position = index;
    notifyListeners();
  }

  void nodeNotFound() {
    _isSearching = false;
    _position = -1;
    notifyListeners();
  }
}
```

Class SearchProvider được kế thừa từ BaseProvider được khai báo mặc định các giá trị trong mảng. Biến isSearching dùng để ngăn sự kiện của nút Search trên màn hình, nếu isSearching bằng true khi mà thuật toán đang chạy thì không thể tìm kiếm được, ngược lại thì có thể tìm kiếm bằng một giá trị khác.

Hàm reset: Dùng để đặt lại giá trị trong mảng.

Hàm potentialNode: Dùng để thiết lập khi mà giá trị vào hàm đó.

Đối với hàm searchedNode, discardNode, foundNode, nodeNotFound cũng tương tự.

Hàm discardNodes: khác với hàm discardNode là nó dùng đối với thuật toán Binary search dùng để biến một dãy các giá trị bên trái hoặc bên phải chứ không phải từng giá trị.

2.3. Cài đặt thuật toán Binary search

```dart
import 'package:algorithm/models/search_model.dart'; // lấy file seach_model.dart
import 'package:algorithm/providers/search/search_provider.dart'; // lấy file seach_model.dart

class BinarySearchProvider extends SearchProvider {
  @override
  void search({int value = 34}) {
    super.search();
    _startBinarySearch(numbers, value);
  }

  Future<int> _startBinarySearch(List<SearchModel> list, int target) async {
    return _binarySearchHelper(list, target, 0, list.length - 1);
  }

  Future<int> _binarySearchHelper(
      List<SearchModel> list, int target, int left, int right) async {
    while (left <= right) {
      final middle = (left + right) ~/ 2; // lấy vị trí giữa mảng
      potentialNode(
          middle); // Dùng để thiết lập khi mà giá trị ở vị trí middle vào hàm đó
      await pause(); // Dừng một khoảng thời gian nhất định
      final potentialMatch = list[middle].value; // Gán giá trị vào biến
      if (target == potentialMatch) {
        // nếu giá trị bằng giá trị cuối mảng
        foundNode(middle); // tìm thấy
        return middle; // trả về vị trí của giá trị
      } else if (target < potentialMatch) {
        discardNodes(middle + 1, right); // bỏ các giá trị bên phải của middle
        await pause();

        right = middle - 1; // thiết lập right bằng middle - 1
      } else {
        discardNodes(left, middle - 1); // bỏ các giá trị bên trái của middle
        await pause();

        left = middle + 1; // thiết lập left bằng middle - 1
      }
      searchedNode(middle);
    }

    nodeNotFound();
    return -1; // not found
  }
}
```

2.4. Cài đặt thuật toán Linear search

```dart
import 'package:algorithm/models/search_model.dart'; // lấy file search_model.dart
import 'package:algorithm/providers/search/search_provider.dart'; // lấy file search_provider.dart

class LinearSearchProvider extends SearchProvider {
  @override
  void search({int value = 34}) {
    super.search(value: value);
    _startSearch(numbers, value);
  }

  Future _startSearch(List<SearchModel> list, int target) async {
    // duyệt các phần tử trong mảng
    for (var index = 0; index < list.length; index++) {
      potentialNode(
          index); // Dùng để thiết lập khi mà giá trị ở vị trí index vào hàm đó
      await pause(); // Dừng một khoảng thời gian nhất định
      if (numbers[index].value == target) {
        // nếu giá trị ở vị trí bằng giá trị cần tìm
        foundNode(index); // tìm thấy giá trị
        return; // thoát khỏi vòng lặp
      } else {
        // nếu giá trị ở vị trí không bằng giá trị cần tìm

        discardNode(index); // bỏ  giá trị bên phải của middle
      }
    }
    // duyệt hết mảng mà không thấy
    nodeNotFound(); // trả về not found
  }
}
```

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3.1. Tìm kiếm nhị phân

Link video thực hiện chương trình thuật toán tìm kiếm nhị phân:
https://youtube.com/shorts/6T3r-zKFbxY?feature=share

3.2. Tìm kiếm tuyến tính

Link video thực hiện chương trình thuật toán tìm kiếm tuyến tính:
https://youtube.com/shorts/mOUfrkr0WXE?feature=share

3.3. File apk

Link download file apk:
https://drive.google.com/file/d/10X3_60C7nlVWdZRk87d_h9fer9vYcEq9/view?usp=sharing

TÀI LIỆU THAM KHẢO

1. https://freetuts.net/thuat-toan-tim-kiem-nhi-phan-2634.html
2. Giáo trình CẤU TRÚC DỮ LIỆU – thầy Nguyễn Đức Thuần
3. https://docs.flutter.dev/get-started/install/windows
4. https://dnmtechs.com/thuat-toan-tim-kiem-tuyen-tinh-tim-kiem-tuan-tu/