1. Cài đặt IndexMaxPQ
2. Thành phần dữ liệu và các phương thức:

* Thành phần dữ liệu:
* n: số lượng phần tử trong hàng đợi.
* pq: mảng chứa chỉ mục các phần tử trong binary heap.
* qp: mảng ngược của pq để truy cập trực tiếp chỉ mục.
* keys: mảng chứa các khóa tương ứng với chỉ mục.
* Phương thức:
  + - * IndexMaxPQ(int maxN): Khởi tạo hàng đợi rỗng với chỉ mục từ 0 đến maxN-1.
      * isEmpty(): Kiểm tra hàng đợi rỗng.
      * contains(int i): Kiểm tra tồn tại chỉ mục i.
      * size(): Lấy số lượng phần tử.
      * insert(int i, Key key): Thêm phần tử mới với chỉ mục i và khóa key.
      * maxIndex(): Lấy chỉ mục phần tử có khóa lớn nhất.
      * maxKey(): Lấy khóa lớn nhất.
      * delMax(): Xóa và trả về chỉ mục phần tử khóa lớn nhất.
      * keyOf(int i): Lấy khóa của phần tử tại chỉ mục i.
      * changeKey(int i, Key key): Thay đổi khóa của i thành key.
      * increaseKey(int i, Key key): Tăng khóa của i lên key (key > khóa hiện tại).
      * decreaseKey(int i, Key key): Giảm khóa của i xuống key (key < khóa hiện tại).
      * delete(int i): Xóa phần tử tại chỉ mục.
      * iterator(): Trả về iterator duyệt chỉ mục theo thứ tự giảm của khóa.
      * less(int i, int j): So sánh khóa tại i và j.
      * exch(int i, int j): Hoán đổi phần tử tại i và j trong pq, qp.
      * swim(int k): Đảm bảo tính chất heap sau khi thêm tại k.
      * sink(int k): Đảm bảo tính chất heap sau khi xóa/cập nhật tại k.
* Lớp con HeapIterator
  + - * Thành phần dữ liệu:
* copy: Bản sao của hàng đợi ban đầu.
  + - * Phương thức:
* Constructor: Tạo bản sao của hàng đợi ban đầu vào copy.
* hasNext(): Kiểm tra có phần tử tiếp theo.
* next(): Trả về chỉ mục phần tử tiếp theo.
* remove(): Không implement, ném ngoại lệ.
* main(String[] args): Phương thức chính để kiểm tra.

1. Kết quả:

A close up of a number

Description automatically generated

1. Cài đặt Dijkstra đích.
2. Thành phần dữ liệu và các phương thức:

* Thành phần dữ liệu:
* distOut: mảng lưu trữ khoảng cách ngắn nhất từ đỉnh nguồn đến mỗi đỉnh khác.
* edgeOut: mảng lưu trữ cạnh cuối cùng trên đường đi ngắn nhất từ đỉnh nguồn đến mỗi đỉnh khác.
* pq: hàng đợi ưu tiên (IndexMinPQ) để lựa chọn đỉnh tiếp theo trong thuật toán Dijkstra.
* Phương thức:
* DijkstraSP\_T(EdgeWeightedDigraph\_T G, int s): Khởi tạo đối tượng và tính toán đường đi ngắn nhất từ đỉnh nguồn s đến tất cả các đỉnh khác trong đồ thị G.
* distOut(int v): Trả về khoảng cách ngắn nhất từ đỉnh nguồn đến đỉnh v.
* hasPathOut(int v): Kiểm tra xem có tồn tại đường đi từ đỉnh nguồn đến đỉnh v hay không.
* pathOut(int v): Trả về một bộ duyệt các cạnh trên đường đi ngắn nhất từ đỉnh nguồn đến đỉnh v.
* relax(DirectedEdge e): Cập nhật khoảng cách ngắn nhất khi tìm thấy đường đi ngắn hơn.
* check(EdgeWeightedDigraph\_T G, int s): Kiểm tra tính đúng đắn của kết quả bằng cách kiểm tra các điều kiện tối ưu.
* validateVertex(int v): Kiểm tra tính hợp lệ của một đỉnh.
* main(String[] args): Phương thức chính để chạy ứng dụng, đọc đầu vào từ tệp tin và in ra đường đi ngắn nhất.

1. Kết quả: A number with numbers and symbols

   Description automatically generated with medium confidence
2. Cài đặt DijkstraAllPairs: tìm mọi đường đi giữa hai đỉnh.
3. Thành phần dữ liệu và các phương thức:

* Thành phần dữ liệu:
* all: Một mảng các đối tượng DijkstraSP, mỗi đối tượng tính toán đường đi ngắn nhất từ một đỉnh cho trước đến tất cả các đỉnh khác trong đồ thị.
* Phương thức:
* DijkstraAllPairsSP(EdgeWeightedDigraph G): Khởi tạo một đối tượng DijkstraAllPairsSP cho đồ thị có trọng số G. Phương thức này tạo ra mảng all và khởi tạo mỗi phần tử trong mảng là một đối tượng DijkstraSP tương ứng với một đỉnh khởi đầu khác nhau.
* path(int s, int t): Trả về một danh sách các cạnh trên đường đi ngắn nhất từ đỉnh s đến đỉnh t. Nó sử dụng đối tượng DijkstraSP tương ứng trong mảng all[s] để tìm đường đi.
* hasPath(int s, int t): Kiểm tra xem có đường đi từ đỉnh s đến đỉnh t hay không bằng cách kiểm tra khoảng cách từ s đến t có bằng vô cực hay không.
* dist(int s, int t): Trả về khoảng cách ngắn nhất từ đỉnh s đến đỉnh t. Nó sử dụng đối tượng DijkstraSP tương ứng trong mảng all[s] để tính khoảng cách.
* validateVertex(int v): Phương thức riêng tư để kiểm tra tính hợp lệ của một đỉnh trong đồ thị. Nó ném ra một IllegalArgumentException nếu đỉnh không nằm trong khoảng từ 0 đến số đỉnh của đồ thị trừ đi 1.
* main(String[] args): Phương thức chính để chạy ứng dụng. Nó đọc đồ thị có trọng số từ tệp tin, tạo một đối tượng DijkstraAllPairsSP, và in ra ma trận khoảng cách và đường đi ngắn nhất giữa tất cả các cặp đỉnh.

1. Kết quả:

Ma trận khoảng cách:

A number and numbers on a white background

Description automatically generated

Đường đi ngắn nhất giữa các cặp đỉnh:

A table with numbers and symbols

Description automatically generated

A white paper with black text and numbers

Description automatically generated

A white paper with numbers and symbols

Description automatically generated

1. Cài đặt PrimMST
2. Thành phần dữ liệu và các phương thức:

* Thành phần dữ liệu:
* edgeTo: Mảng các cạnh, edgeTo[v] là cạnh ngắn nhất từ đỉnh thuộc cây đến đỉnh v không thuộc cây.
* distTo: Mảng các khoảng cách, distTo[v] là độ dài của cạnh ngắn nhất đó (edgeTo[v]).
* marked: Mảng boolean, marked[v] là true nếu v thuộc cây, false ngược lại.
* pq: Một hàng đợi ưu tiên IndexMinPQ lưu trữ các đỉnh v và distTo[v].
* Phương thức:
* PrimMST(EdgeWeightedGraph G): Khởi tạo đối tượng và tính toán MST cho đồ thị có trọng số G.
* relax(Edge e, int v): Cập nhật distTo và edgeTo nếu cạnh e có trọng số nhỏ hơn distTo hiện tại của đỉnh kề với v.
* edges(): Trả về một tập hợp các cạnh của MST.
* weight(): Tính và trả về trọng lượng (tổng trọng số của các cạnh) của MST.
* check(EdgeWeightedGraph G): Kiểm tra tính đúng đắn của MST bằng cách kiểm tra các điều kiện tối ưu.
* main(String[] args): Phương thức chính để chạy ứng dụng, đọc đầu vào từ tệp tin và in ra các cạnh của đồ thị ban đầu, MST và trọng lượng của MST.

1. Kết quả:

A screenshot of a computer screen

Description automatically generated

1. Cài đặt Bellman-Ford đích
2. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:

distTo[]: Mảng lưu khoảng cách ngắn nhất từ đỉnh nguồn đến các đỉnh khác.

edgeTo[]: Mảng lưu cạnh cuối cùng trên đường đi ngắn nhất từ đỉnh nguồn đến các đỉnh khác.

onQueue[]: Mảng boolean để đánh dấu đỉnh đang nằm trong hàng đợi.

queue: Hàng đợi chứa các đỉnh cần được cập nhật.

cost: Biến đếm số lần gọi hàm relax().

cycle: Chu trình âm (hoặc null nếu không có chu trình âm).

* Phương thức:

1. Kết quả:

A number with numbers and symbols

Description automatically generated with medium confidence

1. Đóng gói ba lô có phương thức: taobalo(), tonggiatri(), dovat()
2. Đóng gói Nqueens: SolutionN, Nqueens: Tạo đối tượngNQueens(n), InKetqua() /\* in 96 hoan vi 8 phan tu\*/, Insoluongketqua(),
3. Giải thích các chương trình ứng dụng: TopM, SET, FrequencyCounter, BlackList, WhiteList, FileIndex, LookupIndex.
4. Đóng gói: FileFrequencyIndex: các phương thức Đọc file, query(word)
5. Cài đặt lớp Sinh vien Sinhvien: tính diem TBC, tinh diem TBC hoc ky(kythu), NhapdiemmonSV(Mon m, Integer diem)
6. DS Lop: Tao từ file Excel, Nhap diem mon, TinhTBC(), Tổng kết học kỳ (kythu) /\*In 10 sv có TBCkythu do là cao nhat\*/
   * 1. In danh sách lớp sắp xếp theo tên
     2. In danh sách lớp sắp xếp theo ngày sinh từ già đến trẻ
     3. In danh sách sinh viên theo Quê (các sinh viên cùng quê sắp xếp theo tên). Cho quê, đưa ra danh sách sinh viên có quê đó.
7. TopM sinh vien: theo tieu chi Trẻ, Điểm TBC, Tên và họ đệm
8. DSSinhvien dùng Bảng băm HashMaps: HashMap<Student, MaSV>: gan mã sinh viên cho sinh viên khi nhập học: Doc file danh sach sinh vien tạo Mảng Sinhvien, rồi sắp xếp theo thứ tự gồm tên, họ đệm, ngày sinh sau đó dùng mảng đã sắp xếp put(sinh vien, masv) vào bảng băm Hashmap, trong đó mã sinh vien tăng dần theo ký hiệu của Trường.
9. Giả sử cho danh sách sinh vien dang excel: sinhvien.csv, trong đó có các cột masv, Tên, họ đệm, Điểm TBC, ngày tháng năm sinh, quê quán (tỉnh). Đọc file tạo đối tượng DSLop và cài đặt thêm các phương thức
   1. Truy vấn: InDanhsachĐồnghương (tỉnh) đưa ra các sinh viên có quê là tỉnh đã cho.
   2. Truy vấn: IndanhsachSinhvien(diem) đưa ra các sinh viên có điểm TBC từ cao đến thấp, mà điểm TBC >= diem
10. Cài đặt Ford-Fulkerson-Cut
11. VietnamDate: bổ sung nhiều cách tạo đối tượng Date: ngày/tháng/ năm; ngày - tháng – năm, ngày … tháng … năm …
12. Đóng gói Inversion trong Project Beyond: Cho một file Text danh sách sinh viên, mỗi sinh viên 1 dòng. Danh sách sinh viên trên chưa được sắp xếp theo (tên, họ và đệm). Tìm số nghịch thể trong danh sách đã cho và sắp xếp lại theo thứ tự tên, họ đệm
13. Cài đặt tính toán biểu thức hậu tố có bổ sung phép chia số thực. Thêm phép chia vào SimpleSuffix trong Project BagQueueStack
14. Dựa trên Closest Pairs trong Project Beyond cài đặt FarthestPair: tìm 2 diểm xa nhau nhất.
15. Cài đặt thuật toán Erliest-Finish-Time-First cho bài toán Interval Scheduling trong Bài giảng Greedy Algorithm 1.
16. Đóng gói xếp lịch Weighted Interval Scheduling: đọc trong bài giảng Dynamic programming 1 và dựa trên file Job trong Context.
17. FileSearch: FileFrequencyIndex; File-Count: File, Count (so sánh); Query: chứa từ khóa words [], results <File-Count>.
18. Cài đặt Edmond-Branching.