TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------



**BÁO CÁO CUỐI KỲ**

**PHÂN TÍCH THIẾT KẾ THUẬT TOÁN**

Giảng viên hướng dẫn: Trần Văn Dũng

Sinh viên thực hiện: Đào Minh Quân

MSV: 221230966

Lớp: CNTT 2

Khóa: K63

**Hà Nội, tháng 5 năm 2024**

**Các bài chưa làm được: 9, 13, 15, 17, 18 , 19, 20, 21, 22 ,23**

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

* Thành phần dữ liệu:
* n: số lượng phần tử có 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.
      * 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.

*Độ phức tạp: O(log n)*

* + - * maxIndex(): Lấy chỉ mục phần tử có khóa lớn nhất.

*Độ phức tạp: O(1).*

* + - * maxKey(): Lấy khóa lớn nhất.

*Độ phức tạp: O(1).*

* + - * delMax(): Xóa và trả về chỉ mục phần tử khóa lớn nhất.

*Độ phức tạp: O(log n).*

* + - * keyOf(int i): Lấy khóa của phần tử tại chỉ mục i.

*Độ phức tạp: O(1).*

* + - * changeKey(int i, Key key): Thay đổi khóa của i thành key.

*Độ phức tạp: O(log n).*

* + - * increaseKey(int i, Key key): Tăng khóa của i lên key (key > khóa hiện tại).

*Độ phức tạp: O(log n).*

* + - * decreaseKey(int i, Key key): Giảm khóa của i xuống key (key < khóa hiện tại).

*Độ phức tạp: O(log n).*

* + - * delete(int i): Xóa phần tử tại chỉ mục.

*Độ phức tạp: O(log n).*

* + - * iterator(): Trả về iterator duyệt chỉ mục theo thứ tự giảm của khóa.

*Độ phức tạp: O(n).*

* + - * 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:

9 worst

1 was

7 was

5 times

8 the

2 the

4 of

6 it

0 it

3 best

7 waswas

9 w

1 w

8 thethe

5 t

2 t

4 o

6 itit

0 i

3 b

0 it

2 the

9 worst

1 was

8 the

7 was

4 of

6 it

3 best

5 times

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

* Thành phần dữ liệu:
* distTo: mảng lưu trữ khoảng cách ngắn nhất từ đỉnh nguồn đến mỗi đỉnh chỉ số.
* edgeTo: mảng lưu trữ cạnh cuối cùng trên đường đi ngắn nhất từ đỉnh nguồn đến mỗi đỉnh chỉ số.
* pq: hàng đợi ưu tiên (IndexMinPQ) để lựa chọn đỉnh tiếp theo trong thuật toán Dijkstra.
* Phương thức:
* Dijkstra\_T (EdgeWeightedDigraph\_T G, int s): Khởi tạo đối tượng và tính toán đường đi ngắn nhất từ đỉnh tất cả các đỉnh đỉnh nguồn s trong đồ thị G.

Độ phức tạp: O(nlogn).

* distTo (int v): Trả về khoảng cách ngắn nhất từ đỉnh v đến đỉnh nguồn.

Độ phức tạp: O(1).

* hasPathTo(int v): Kiểm tra xem có tồn tại đường đi từ đỉnh v đến đỉnh nguồn hay không.

Độ phức tạp: O(1).

* pathTo(int v): Trả về một bộ duyệt các cạnh trên đường đi ngắn nhất từ đỉnh v đến đỉnh nguồn.

Độ phức tạp: O(n).

* relax(DirectedEdge e): Cập nhật khoảng cách ngắn nhất khi tìm thấy đường đi tốt hơn.

Độ phức tạp: O(logn).

* check(EdgeWeightedDigraph\_T G, int s): Kiểm tra tính đúng đắn của kết quả bằng cách kiểm tra các điều kiện tối ưu.

Độ phức tạp: O(nlogn),

* validateVertex(int v): Kiểm tra tính hợp lệ của một đỉnh.

Độ phức tạp: O(1).

* main(String[] args): Phương thức chính để chạy ứng dụng, đọc đầu vào từ tệp tin và in ra đường đi ngắn nhất.

Kết quả:

("tinyEWD.txt","0"):

0 to 0 (0.00)

1 to 0 (1.39) 1->3 0,29 3->6 0,52 6->0 0,58

2 to 0 (1.83) 2->7 0,34 7->3 0,39 3->6 0,52 6->0 0,58

3 to 0 (1.10) 3->6 0,52 6->0 0,58

4 to 0 (1.86) 4->7 0,37 7->3 0,39 3->6 0,52 6->0 0,58

5 to 0 (1.71) 5->1 0,32 1->3 0,29 3->6 0,52 6->0 0,58

6 to 0 (0.58) 6->0 0,58

7 to 0 (1.49) 7->3 0,39 3->6 0,52 6->0 0,58

1. **Cài đặt DijkstraAllPairs: tìm mọi đường đi giữa hai đỉnh.**
2. 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.

Độ phức tạp: O(nlogn)

* 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.

Độ phức tạp: O(n)

* 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.

Độ phức tạp: O(1)

* 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.

Độ phức tạp: O(1)

* 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.

Độ phức tạp: O(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.

Kết quả:

{"tinyEWG.txt"}

0 1 2 3 4 5 6 7

0: 0.00 Inf 0.26 0.43 0.38 0.73 0.95 0.16

1: 1.39 0.00 0.36 0.29 1.74 0.32 0.81 0.19

2: 1.27 Inf 0.00 0.17 1.62 1.97 0.69 0.34

3: 1.10 Inf 0.92 0.00 1.45 1.80 0.52 1.26

4: Inf Inf Inf Inf 0.00 0.35 Inf 0.37

5: Inf Inf Inf Inf Inf 0.00 Inf 0.28

6: 0.58 Inf 0.40 0.57 0.93 1.28 0.00 0.74

7: Inf Inf Inf Inf Inf Inf Inf 0.00

0 to 0 ( 0.00)

0 to 1 no path

0 to 2 ( 0.26) 0->2 0,26

0 to 3 ( 0.43) 0->2 0,26 2->3 0,17

0 to 4 ( 0.38) 0->4 0,38

0 to 5 ( 0.73) 0->4 0,38 4->5 0,35

0 to 6 ( 0.95) 0->2 0,26 2->3 0,17 3->6 0,52

0 to 7 ( 0.16) 0->7 0,16

1 to 0 ( 1.39) 1->3 0,29 3->6 0,52 6->0 0,58

1 to 1 ( 0.00)

1 to 2 ( 0.36) 1->2 0,36

1 to 3 ( 0.29) 1->3 0,29

1 to 4 ( 1.74) 1->3 0,29 3->6 0,52 6->4 0,93

1 to 5 ( 0.32) 1->5 0,32

1 to 6 ( 0.81) 1->3 0,29 3->6 0,52

1 to 7 ( 0.19) 1->7 0,19

2 to 0 ( 1.27) 2->3 0,17 3->6 0,52 6->0 0,58

2 to 1 no path

2 to 2 ( 0.00)

2 to 3 ( 0.17) 2->3 0,17

2 to 4 ( 1.62) 2->3 0,17 3->6 0,52 6->4 0,93

2 to 5 ( 1.97) 2->3 0,17 3->6 0,52 6->4 0,93 4->5 0,35

2 to 6 ( 0.69) 2->3 0,17 3->6 0,52

2 to 7 ( 0.34) 2->7 0,34

3 to 0 ( 1.10) 3->6 0,52 6->0 0,58

3 to 1 no path

3 to 2 ( 0.92) 3->6 0,52 6->2 0,40

3 to 3 ( 0.00)

3 to 4 ( 1.45) 3->6 0,52 6->4 0,93

3 to 5 ( 1.80) 3->6 0,52 6->4 0,93 4->5 0,35

3 to 6 ( 0.52) 3->6 0,52

3 to 7 ( 1.26) 3->6 0,52 6->2 0,40 2->7 0,34

4 to 0 no path

4 to 1 no path

4 to 2 no path

4 to 3 no path

4 to 4 ( 0.00)

4 to 5 ( 0.35) 4->5 0,35

4 to 6 no path

4 to 7 ( 0.37) 4->7 0,37

5 to 0 no path

5 to 1 no path

5 to 2 no path

5 to 3 no path

5 to 4 no path

5 to 5 ( 0.00)

5 to 6 no path

5 to 7 ( 0.28) 5->7 0,28

6 to 0 ( 0.58) 6->0 0,58

6 to 1 no path

6 to 2 ( 0.40) 6->2 0,40

6 to 3 ( 0.57) 6->2 0,40 2->3 0,17

6 to 4 ( 0.93) 6->4 0,93

6 to 5 ( 1.28) 6->4 0,93 4->5 0,35

6 to 6 ( 0.00)

6 to 7 ( 0.74) 6->2 0,40 2->7 0,34

7 to 0 no path

7 to 1 no path

7 to 2 no path

7 to 3 no path

7 to 4 no path

7 to 5 no path

7 to 6 no path

7 to 7 ( 0.00)

**4. Cài đặt PrimMST**

1. Thành phần dữ liệu và các phương thức:

* Thành phần dữ liệu:
* edgeTo: Mảng các cạnh, edgeTo[v] là cạnh có trọng số nhỏ nhất từ đỉnh thuộc cây đến đỉnh v không thuộc cây.
* distTo: Mảng các khoảng cách, distTo[v] là độ dài của cạnh có trọng số nhỏ 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 độ ưu tiên distTo[v].
* Phương thức:
* PrimMST(EdgeWeightedGraph G): Khởi tạo đối tượng và tính toán MST cho đồ thị có trọng số G.

Độ phức tạp: O(nlogn),

* prim(EdgeWeightedGraph G, int s): Chạy thuật toán bắt đầu từ đỉnh s, liên tục lấy đỉnh có khoảng cách nhỏ nhất từ hàng đợi ưu tiên và quét các đỉnh kề của nó.

Độ phức tạp: O(nlogn),

* scan(EdgeWeightedGraph G, int v): Quét và cập nhật các đỉnh kề với đỉnh v.

Độ phức tạp: O(nlogn),

* edges(): Trả về một tập hợp các cạnh của MST.

Độ phức tạp: O(n),

* weight(): Tính và trả về trọng lượng (tổng trọng số của các cạnh) của MST.

Độ phức tạp: O(n),

* check(EdgeWeightedGraph G): Kiểm tra tính đúng đắn của MST bằng cách kiểm tra các điều kiện tối ưu.

Độ phức tạp: O(nlogn),

* main(String[] args): Phương thức chính để chạy ứng dụng, đọc đầu vào từ tệp tin và Tính toán cây MST. In ra các cạnh và tổng trọng số của cây MST.

1. Kết quả:

{"tinyEWG.txt", "0"}

1-7 0,19000

0-2 0,26000

2-3 0,17000

4-5 0,35000

5-7 0,28000

6-2 0,40000

0-7 0,16000

1.81000

**5. Cài đặt Bellman-Ford đích**

1. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* distFrom[]: Mảng lưu trữ khoảng cách từ đỉnh nguồn đến mỗi đỉnh khác. distFrom[v] là khoảng cách ngắn nhất từ đỉnh nguồn đến đỉnh v.
* edgeFrom[]: Mảng lưu trữ các cạnh trên đường đi ngắn nhất từ đỉnh nguồn đến mỗi đỉnh khác. edgeFrom[v] là cạnh cuối cùng trên đường đi ngắn nhất từ đỉnh nguồn đến đỉnh v.
* onQueue[]: Mảng boolean xác định xem một đỉnh có đang được xem xét trong hàng đợi không. onQueue[v] là true nếu đỉnh v đang có trong hàng đợi.
* queue: Hàng đợi chứa các đỉnh cần được xem xét để thả lỏng. Các đỉnh này được thêm vào hàng đợi khi cần thả lỏng và được loại bỏ khi đã xử lý xong.
* cost: Số lượng cuộc gọi cho phương thức thả lỏng. Sử dụng để kiểm tra xem có chu trình âm hay không.
* cycle: Số lượng cuộc gọi cho phương thức thả lỏng. Sử dụng để kiểm tra xem có chu trình âm hay không.
* Phương thức:
* BellmanFord\_T(EdgeWeightedDigraph\_T G, int s): Constructor của lớp, tính toán đường đi ngắn nhất từ mỗi đỉnh đến đỉnh nguồn s trong đồ thị có trọng số G.

Độ phức tạp: O(n^2)

* relax(EdgeWeightedDigraph\_T G, int v): Thả lỏng đỉnh v và đặt các đỉnh kết thúc khác vào hàng đợi nếu có thay đổi.

Độ phức tạp: O(n)

* hasNegativeCycle(): Kiểm tra xem có chu trình âm từ đỉnh nguồn hay không.

Độ phức tạp: O(1)

* negativeCycle(): Trả về chu trình âm, nếu có.

Độ phức tạp: O(1)

* findNegativeCycle(): Tìm chu trình âm trong đồ thị bằng cách tạo một đồ thị con từ các cạnh edgeFrom và sử dụng một thuật toán phát hiện chu trình.
* Độ phức tạp: O(n^2)
* distFrom(int v): Trả về khoảng cách ngắn nhất từ đỉnh nguồn đến đỉnh v.

Độ phức tạp: O(1)

* hasPathFrom(int v): Kiểm tra xem có đường đi từ đỉnh nguồn đến đỉnh v hay không.

Độ phức tạp: O(1)

* pathTo(int v): Trả về đường đi ngắn nhất từ đỉnh nguồn đến đỉnh v dưới dạng một ngăn xếp các cạnh.

Độ phức tạp: O(n)

* check(EdgeWeightedDigraph\_T G, int s): Kiểm tra các điều kiện tối ưu: có chu trình âm từ đỉnh nguồn hoặc mọi cạnh đều thoả mãn điều kiện đường đi ngắn nhất.

Độ phức tạp: O(n^2)

* validateVertex(int v): Ném ngoại lệ nếu chỉ số đỉnh không hợp lệ.

Độ phức tạp: O(1)

* main(String[] args): Phương thức main để thực hiện các bài kiểm tra và in kết quả.

1. Kết quả:

{"tinyEWDn.txt", "0"}

Satisfies optimality conditions

0 to 0 ( 0.00)

1 to 0 (-0.59) 6->0 -1,40 3->6 0,52 1->3 0,29

2 to 0 (-0.15) 6->0 -1,40 3->6 0,52 7->3 0,39 2->7 0,34

3 to 0 (-0.88) 6->0 -1,40 3->6 0,52

4 to 0 (-0.12) 6->0 -1,40 3->6 0,52 7->3 0,39 4->7 0,37

5 to 0 (-0.27) 6->0 -1,40 3->6 0,52 1->3 0,29 5->1 0,32

6 to 0 (-1.40) 6->0 -1,40

7 to 0 (-0.49) 6->0 -1,40 3->6 0,52 7->3 0,39

**6. Đóng gói ba lô có phương thức: taobalo(), tonggiatri(), dovat()**

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class **Item** cung cấp các phương thức getter để lấy thông tin về tên, trọng lượng và giá trị của mặt hàng.
* Class **KnapsackImplementation** tính toán và lưu trữ các thông tin liên quan đến việc chọn mặt hàng để đặt vào túi, bao gồm trọng lượng thực tế, tổng giá trị và danh sách các mặt hàng đã chọn.
* Phương thức:
* Class này là nơi chạy chương trình, tạo ra các mặt hàng ngẫu nhiên và thực hiện giải quyết bài toán cái túi bằng cách sử dụng lớp **KnapsackImplementation.**

b. Kết quả:

{"6", "2000"}

List items:

Name: item1 weight: 1167 Value: 771

Name: item2 weight: 378 Value: 9

Name: item3 weight: 1708 Value: 543

Name: item4 weight: 1125 Value: 539

Name: item5 weight: 1834 Value: 604

Name: item6 weight: 1756 Value: 743

total value: 780

Name: item2 weight: 378 Value: 9

Name: item1 weight: 1167 Value: 771

real weight: 1545

max weight: 2000

**7. Đóng gói Nqueens: SolutionN, Nqueens: Tạo đối tượngNQueens(n), InKetqua() /\* in 96 hoan vi 8 phan tu\*/, Insoluongketqua(),**

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class SolutionN: Lưu trữ một giải pháp của bài toán N-Queens.

+, board: Một mảng hai chiều đại diện cho bảng cờ của giải pháp, trong đó giá trị 1 đại diện cho quân hậu được đặt và 0 đại diện cho ô trống.

Phương thức:

+, SolutionN(int[][] board): Constructor để khởi tạo một đối tượng SolutionN với bảng cờ được đưa vào.

+, getBoard(): Trả về bảng cờ của giải pháp.

+, setBoard(int[][] board): Đặt bảng cờ cho giải pháp.

+, printBoard(): In ra bảng cờ của giải pháp.

* Class AllSolNQueen: Tìm tất cả các giải pháp của bài toán

N-Queens.

+, N: Số quân hậu cần đặt trên bàn cờ.

+, solutions: Danh sách chứa tất cả các giải pháp của bài toán.

Phương thức:

+, AllSolNQueen(int N): Constructor để khởi tạo một đối tượng AllSolNQueen với số quân hậu là N, tìm tất cả các giải pháp và lưu chúng trong danh sách solutions.

+, SolveNQ(): Phương thức private thực hiện tìm tất cả các giải pháp của bài toán.

+, isSafe(int board[][], int row, int col): Phương thức private kiểm tra xem việc đặt quân hậu tại vị trí (row, col) có an toàn không.

+, solveNQUtil(int board[][], int col): Phương thức private sử dụng phương pháp quay lui để tìm các giải pháp của bài toán.

+, printNSolution(int n): In ra giải pháp thứ n.

+, printAllSolutions(): In ra tất cả các giải pháp.

+, getResults(): Trả về số lượng giải pháp tìm được.

* Class NQueen: Lớp chính để chạy chương trình và kiểm tra tất cả các giải pháp của bài toán N-Queens.

+, main(String args[]): Phương thức chính để chạy chương trình. Tạo một đối tượng AllSolNQueen và in ra tất cả các giải pháp của bài toán.

b. Kết quả:

Complete!

Total of result: 92

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

**8. Giải thích các chương trình ứng dụng: TopM, SET, FrequencyCounter, BlackList, WhiteList, FileIndex, LookupIndex.**

- TopM : Tìm ra m giao dịch (Transaction) lớn nhất. TopM sử dụng hàng đợi ưu tiên MinPQ để lưu trữ m giao dịch từ nhỏ đến lớn nhất. Sau đó push lần lượt vào một Stack để lấy ra từ lớn nhất đến thấp nhất.

- SET : sử dụng kiểu dữ liệu TreeMap để lưu trữ những key.

- FrequencyCounter : Sử dụng kiểu dữ liệu cây tìm kiếm Symbol Table (ST) để lưu trữ những từ có độ dài lớn hơn ngưỡng cho trước, trong đó key : là từ đó, còn value : là số lần xuất hiện.

- BlackFilter : Sử dụng kiểu dữ liệu SET (TreeMap) để lưu trữ một danh sách các từ cấm. Đọc những từ từ một file , nếu không phải từ cấm (không thuộc Set) thì in ra.

- WhiteFilter : In ra những từ được cho phép từ trong file. Danh sách những từ được cho phép được lưu trữ trong một Set (tương tự BlackFilter).

- FileIndex : có thuộc tính là một Symbol Table có key là từ, value là một Set các file chứa từ đó. FileIndex sẽ in ra những file chứa từ cần tìm.

- LookupIndex : là một chương trình để tạo một chỉ mục (index) dựa trên các cặp key - value từ một tệp tin. Sau đó, nó cho phép người dùng nhập các từ và in ra các value tương ứng với các key được tìm thấy. Những key – value được lưu trữ bằng Symbol Table với cặp key, và queue những value tương ứng với nó.

**10. Cài đặt lớp Sinh vien Sinhvien: tính diem TBC, tinh diem TBC hoc ky(kythu), NhapdiemmonSV(Mon m, Integer diem)**

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class **Mon** đóng vai trò để lưu trữ thông tin về một môn học cụ thể, bao gồm tên môn và điểm số.
* Class **VNDate** được thiết kế để biểu diễn và thao tác với các ngày tháng trong lịch Việt Nam.
* ten: lưu trữ tên của sinh viên
* ngaySinh: lưu trữ ngày sinh của sinh viên
* queQuan: Lưu trữ quê quán của sinh viên.
* diemMonHoc: Lưu trữ danh sách các môn học mà sinh viên đã tham gia và điểm số tương ứng.
* Phương thức:
* SinhVien(String ten, VNDate ngaySinh, String queQuan): Tạo đối tượng SinhVien với các thông tin cơ bản như tên, ngày sinh và quê quán.
* tinhDiemTBC(): Tính điểm trung bình của tất cả các môn học mà sinh viên đã tham gia.
* tinhDiemTBCHocKy(String kyThu): Tính điểm trung bình của các môn học trong một kỳ học cụ thể.
* main(String[] args): Đọc dữ liệu từ một tệp văn bản (sinhvien.txt), tạo đối tượng SinhVien từ dữ liệu và tính toán điểm trung bình.

In ra thông tin của mỗi sinh viên và điểm trung bình của họ.

b. Kết quả:

Tên: Nguyen Van X

Ngày sinh: 10/8/1997

Quê quán: Hà Tĩnh

Điểm trung bình: 8.3

Không có dữ liệu cho kỳ 1

Điểm trung bình của kỳ 2: 8.0

Điểm trung bình của kỳ 3: 8.0

Tên: Tran Thi Y

Ngày sinh: 5/3/2001

Quê quán: Hải Phòng

Điểm trung bình: 8.2

Điểm trung bình của kỳ 1: 8.5

Điểm trung bình của kỳ 2: 7.5

Điểm trung bình của kỳ 3: 8.0

Tên: Le Van Z

Ngày sinh: 20/11/1999

Quê quán: Quảng Nam

Điểm trung bình: 8.2

Điểm trung bình của kỳ 1: 8.0

Tên: Pham Thi M

Ngày sinh: 15/9/2002

Quê quán: Bình Định

Điểm trung bình: 7.8

Điểm trung bình của kỳ 1: 7.666666666666667

Điểm trung bình của kỳ 2: 7.5

Tên: Hoang Van N

Ngày sinh: 30/12/2000

Quê quán: Nghệ An

Điểm trung bình: 8.7

Điểm trung bình của kỳ 1: 8.333333333333334

Điểm trung bình của kỳ 2: 8.5

**11.** **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.

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class **Mon** được thiết kế để biểu diễn thông tin về một môn học, bao gồm tên và số tín chỉ.
* Class **VietnamDate** là một lớp không thay đổi, đại diện cho một ngày cụ thể với các thuộc tính ngày, tháng và năm.
* Class **Sinhvien** là một lớp chứa thông tin chi tiết về sinh viên, bao gồm mã sinh viên, họ tên, ngày sinh, điểm trung bình, quê quán, số điện thoại và bảng điểm của sinh viên. Lớp này cung cấp các phương thức để khởi tạo đối tượng Sinhvien, lấy thông tin sinh viên, tính điểm trung bình, và so sánh sinh viên theo tên hoặc điểm trung bình
* Class **ST** là một bảng ký hiệu được sắp xếp sử dụng TreeMap, hỗ trợ các thao tác cơ bản như chèn, xóa, tìm kiếm, kiểm tra kích thước, và duyệt qua các khóa
* Phương thức:
* Chương trình này đọc dữ liệu sinh viên từ tập tin CSV, tính toán và lưu trữ điểm trung bình của mỗi sinh viên, sắp xếp danh sách sinh viên theo nhiều tiêu chí khác nhau và in ra kết quả theo các yêu cầu cụ thể.

b. Kết quả:

Top 10 sinh vien co diem trung binh ky cao nhat:

2 Do Xuan KiBS 5/8/2003 8,10 Hung Yen 2222222

3 Phung Gia NghiAen 26/2/2003 8,10 Ninh Binh 3333333

4 Nguyen Van Hoang 26/2/2003 8,10 Hung Yen 4444444

5 Ta Quoc Viet 26/2/2003 8,10 Thai Binh 5555555

6 Pham Ngoc Khanh 26/2/2003 8,10 Thai Binh 6666666

7 Tong Hai Duong 26/2/2003 8,10 Ninh Binh 7777777

8 Bui Duc Duy 26/2/2003 8,10 Hung Yen 8888888

9 Nguyen Trung Dung 26/2/2003 8,10 Bac Ninh 9999999

1 Nguyen Thanh DatAB 26/2/2003 8,00 Ha Noi 1111111

Danh sach lop sap xep theo ten:

1 Nguyen Thanh DatAB 26/2/2003 8,00 Ha Noi 1111111

9 Nguyen Trung Dung 26/2/2003 8,10 Bac Ninh 9999999

7 Tong Hai Duong 26/2/2003 8,10 Ninh Binh 7777777

8 Bui Duc Duy 26/2/2003 8,10 Hung Yen 8888888

4 Nguyen Van Hoang 26/2/2003 8,10 Hung Yen 4444444

6 Pham Ngoc Khanh 26/2/2003 8,10 Thai Binh 6666666

2 Do Xuan KiBS 5/8/2003 8,10 Hung Yen 2222222

3 Phung Gia NghiAen 26/2/2003 8,10 Ninh Binh 3333333

5 Ta Quoc Viet 26/2/2003 8,10 Thai Binh 5555555

Danh sach lop sap xep theo ngay sinh tu gia den tre:

1 Nguyen Thanh DatAB 26/2/2003 8,00 Ha Noi 1111111

9 Nguyen Trung Dung 26/2/2003 8,10 Bac Ninh 9999999

7 Tong Hai Duong 26/2/2003 8,10 Ninh Binh 7777777

8 Bui Duc Duy 26/2/2003 8,10 Hung Yen 8888888

4 Nguyen Van Hoang 26/2/2003 8,10 Hung Yen 4444444

6 Pham Ngoc Khanh 26/2/2003 8,10 Thai Binh 6666666

3 Phung Gia NghiAen 26/2/2003 8,10 Ninh Binh 3333333

5 Ta Quoc Viet 26/2/2003 8,10 Thai Binh 5555555

2 Do Xuan KiBS 5/8/2003 8,10 Hung Yen 2222222

Danh sach sinh vien theo Que (sap xep theo ten):

Que: Bac Ninh

9 Nguyen Trung Dung 26/2/2003 8,10 Bac Ninh 9999999

Que: Ninh Binh

7 Tong Hai Duong 26/2/2003 8,10 Ninh Binh 7777777

3 Phung Gia NghiAen 26/2/2003 8,10 Ninh Binh 3333333

Que: Ha Noi

1 Nguyen Thanh DatAB 26/2/2003 8,00 Ha Noi 1111111

Que: Hung Yen

8 Bui Duc Duy 26/2/2003 8,10 Hung Yen 8888888

4 Nguyen Van Hoang 26/2/2003 8,10 Hung Yen 4444444

2 Do Xuan KiBS 5/8/2003 8,10 Hung Yen 2222222

Que: Thai Binh

6 Pham Ngoc Khanh 26/2/2003 8,10 Thai Binh 6666666

5 Ta Quoc Viet 26/2/2003 8,10 Thai Binh 5555555

**12. TopM sinh vien: theo tieu chi Trẻ, Điểm TBC, Tên và họ đệm**

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class **student** được sử dụng để đại diện sinh viên.
* Class MinPQ, Stack
* Phương thức:

Class **TopStu** được sử dụng để đọc thông tin sinh viên từ một tập tin và sau đó in ra danh sách các sinh viên có điểm trung bình cộng, ngày sinh, tên hoặc họ tốt nhất hoặc xấu nhất dựa trên số lượng m được chỉ định.

b. Kết quả:

top age

456789 Quy Dijkstra Wed Aug 22 00:00:00 ICT 2007 8,40

234567 Nam Tarjan Tue Mar 26 00:00:00 ICT 2002 6,85

345678 Trung Knuth Mon Jun 14 00:00:00 ICT 1999 8,34

123456 Do Turing Sun Jun 17 00:00:00 ICT 1990 6,08

top tbc

456789 Quy Dijkstra Wed Aug 22 00:00:00 ICT 2007 8,40

345678 Trung Knuth Mon Jun 14 00:00:00 ICT 1999 8,34

234567 Nam Tarjan Tue Mar 26 00:00:00 ICT 2002 6,85

123456 Do Turing Sun Jun 17 00:00:00 ICT 1990 6,08

top name

456789 Quy Dijkstra Wed Aug 22 00:00:00 ICT 2007 8,40

345678 Trung Knuth Mon Jun 14 00:00:00 ICT 1999 8,34

234567 Nam Tarjan Tue Mar 26 00:00:00 ICT 2002 6,85

123456 Do Turing Sun Jun 17 00:00:00 ICT 1990 6,08

top ho

123456 Do Turing Sun Jun 17 00:00:00 ICT 1990 6,08

234567 Nam Tarjan Tue Mar 26 00:00:00 ICT 2002 6,85

456789 Quy Dijkstra Wed Aug 22 00:00:00 ICT 2007 8,40

345678 Trung Knuth Mon Jun 14 00:00:00 ICT 1999 8,34

**14. 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

a. Thành phần dữ liệu và phương thức:

* Thành phần dữ liệu:
* Class **Mon** đại diện cho một môn học
* Class **VietnamDate** là một lớp không thay đổi, đại diện cho một ngày cụ thể với các thuộc tính ngày, tháng và năm.
* Class **Sinhvien** cung cấp các phương thức và thuộc tính để quản lý thông tin của sinh viên và thực hiện các phép toán liên quan đến điểm số và so sánh giữa các sinh viên.
* Class **ST** là một bảng ký hiệu được sắp xếp sử dụng TreeMap, hỗ trợ các thao tác cơ bản như chèn, xóa, tìm kiếm, kiểm tra kích thước, và duyệt qua các khóa
* Phương thức:
* main(String[] arg): Khởi đầu chương trình và gọi các phương thức cần thiết để in ra danh sách sinh viên.
* loadData(): Nạp dữ liệu từ tệp "sinhvien.csv" vào cấu trúc dữ liệu.
* InDanhsachDonghuong(String tinh): In ra danh sách sinh viên có quê quán ở một tỉnh cụ thể.
* IndanhsachSinhvien(double diem): In ra danh sách sinh viên có điểm trung bình lớn hơn hoặc bằng một ngưỡng điểm cho trước.

b. Kết quả:

Danh sach sinh vien que o Ha Noi:

1 Nguyen Thanh Dat 26/2/2003 8,00 Ha Noi 1111111

Danh sach sinh vien co diem trung binh tong >= 8.0:

2 Do Xuan Kien 5/8/2003 8,10 Hung Yen 2222222

3 Phung Gia Nghien 26/2/2003 8,10 Ninh Binh 3333333

4 Nguyen Van Hoang 26/2/2003 8,10 Hung Yen 4444444

5 Ta Quoc Viet 26/2/2003 8,10 Thai Binh 5555555

6 Pham Ngoc Khanh 26/2/2003 8,10 Thai Binh 6666666

7 Tong Hai Duong 26/8/2003 8,10 Ninh Binh 7777777

8 Bui Duc Duy 26/2/2002 8,10 Hung Yen 8888888

9 Nguyen Trung Dung 26/2/2003 8,10 Bac Ninh 9999999

1 Nguyen Thanh Dat 26/2/2003 8,00 Ha Noi 1111111

**16. VietnamDate: bổ sung nhiều cách tạo đối tượng Date: ngày/tháng/ năm; ngày - tháng – năm, ngày … tháng … năm …**

a. Thành phần dữ liệu và phương thức

* Thành phần dữ liệu :
* int day : lưu trữ ngày
* int month : lưu trữ tháng
* int year : lưu trữ năm
* Phương thức :
* VietnamDate(int day, int month, int year) : Hàm tạo đối tượng VietnamDate từ 3 tham số : day, month, year. Nó kiểm tra tính hợp lệ của ngày tháng và ném ra một ngoại lệ nếu không hợp lệ.
* VietnamDate(String date): Hàm tạo cho phép tạo đối tượng VietnamDate từ một chuỗi biểu diễn ngày tháng năm dưới dạng "dd/mm/yyyy". Nó cũng kiểm tra tính hợp lệ của ngày tháng và ném ra một ngoại lệ nếu không hợp lệ.
* parseFromForm(String str): Một phương thức tĩnh để tạo đối tượng VietnamDate từ một chuỗi biểu diễn ngày tháng dưới dạng "ngày - tháng - năm".
* parseFromForm2(String str): Một phương thức tĩnh khác để tạo đối tượng VietnamDate từ một chuỗi biểu diễn ngày tháng dưới dạng "ngày dd tháng mm năm yyyy".
* isValid(int d, int m, int y): Một phương thức tĩnh để kiểm tra tính hợp lệ của một ngày tháng.
* isLeapYear(int y): Một phương thức tĩnh để kiểm tra xem một năm có phải là năm nhuận không.
* next(): Một phương thức để tính toán ngày tiếp theo trong lịch.
* isAfter(VietnamDate that), isBefore(VietnamDate that): Các phương thức để so sánh hai ngày.
* compareTo(VietnamDate that): Phương thức để so sánh hai ngày theo thứ tự thời gian.
* toString(): Phương thức để chuyển đổi một đối tượng VietnamDate thành một chuỗi biểu diễn ngày tháng năm dưới dạng "dd/mm/yyyy"
* equals(Object other): Phương thức để so sánh hai đối tượng VietnamDate xem chúng có bằng nhau không.
* hashCode(): Phương thức để tính mã băm của một đối tượng VietnamDate.

b. Kết quả

Dào Minh Quân

dang .../.../... :

15/5/2024

19/5/1890

30/4/1

1/1/2000

dang ... - ... - ... :

15 - 5 - 2024

19 - 5 - 1890

30 - 4 - 1

1 - 1 - 2000

dang ngay ... thang ... nam ... :

ngay 15 thang 5 nam 2024

ngay 19 thang 5 nam 1890

ngay 30 thang 4 nam 1

ngay 1 thang 1 nam 2000