**Ôn tập**

**Môn Cấu trúc dữ liệu và Giải thuật**

**Phần 1. Danh sách (Danh sách liên kết, danh sách đặc) (4 điểm)**

* Danh sách liên kết: Các thuật toán thêm, xoá, tìm kiếm, tính tổng, đếm các phần tử
* Stack/Queue: thêm phần tử, lấy phẩn tử ra khởi stack/queue
* Các thuật toán sắp xếp: Selection, Bubble, insertion, Merge sort, Quick sort, radix sort: tư tưởng thuật toán, chạy từng bước thuật toán trên mảng cụ thể, đánh giá độ phức tạp của thuật toán.
* Hashing: Thêm phần tử vào hash Table, xử lý đụng độ, tìm kiếm phần tử, xoá phần tử khỏi hash table.

**Phần 2. Cây (3 điểm)**

* Ưu điểm của cây so với Danh sách (liên kết và đặc)
* Cây BST: Dựng cây, duyệt cây, tìm kiếm phần tử, predecessor, successor, tìm max, min, xoá phần tử
* Cây AVL: dựng cây, cân bằng cây, xoá node.
* Cây Binary Max/min Heap: Dựng cây, lấy phần tử max ra khỏi cây, Heapsort

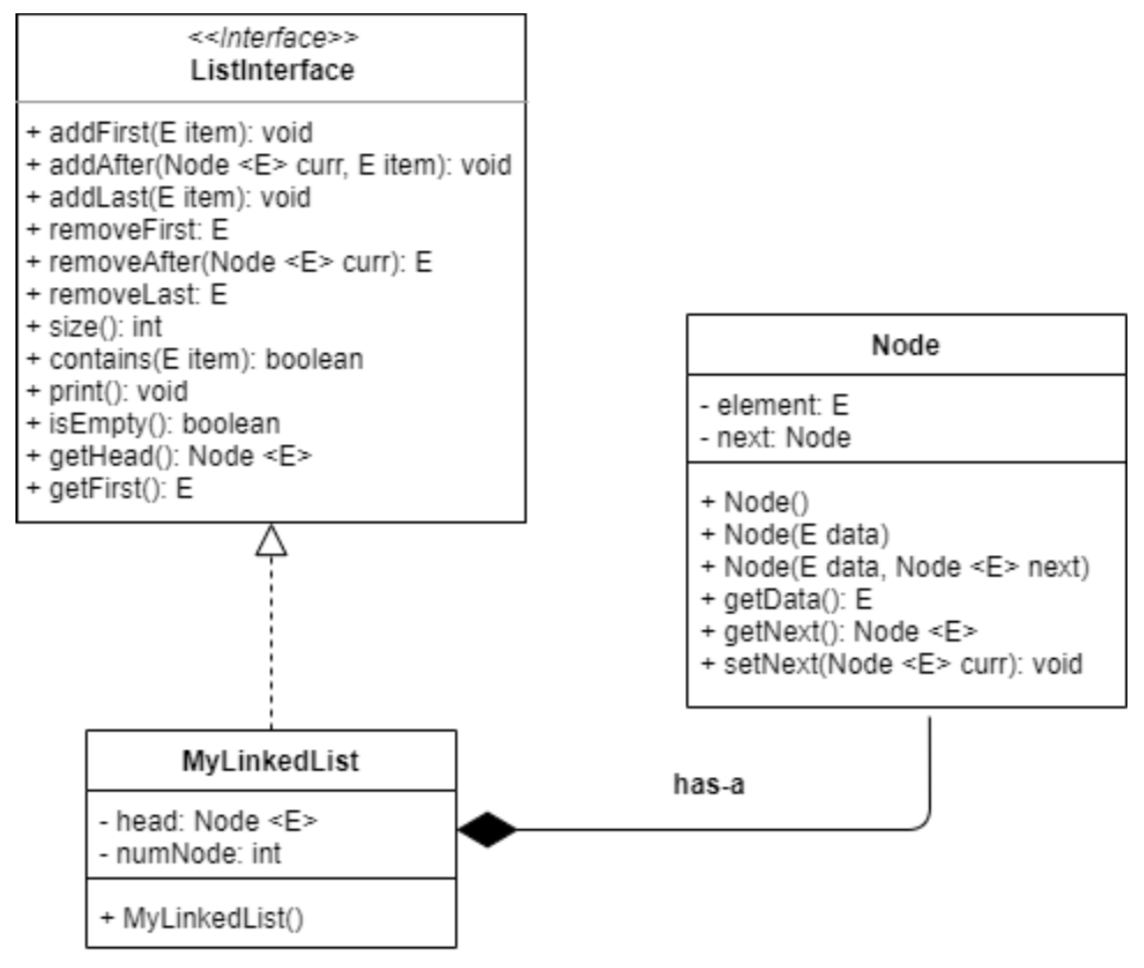
**Phần 3. Đồ thị (3 điểm)**

* Duyệt đồ thị: BFS, DFS
* Cây khung ngắn nhất: Prim, Kruskal
* Đường đi ngắn nhất: Dijkstra, Bellman Ford

**Bài tập ôn**

**Phần 1: Danh sách**

1. Cho sơ đồ lớp Danh sách liên kết như hình:



Viết các phương thức sau:

* Thêm phần tử item vào đầu danh sách liên kết: void addFirst(E item)

Void addFirst(E item)

{

Node p = new Node(item);

p.setNext(head);//p.next = head; head = p;

head = p;

}

* Thêm phần tử item vào sau node hiện hành: void addAfter(Node<E> curr, E item)

Void addAfter(Node<E> curr, E item)

{

Node p = new Node(item);

p.setNext(curr.getNext());//p.next = curr.next

curr.setNext(p); //curr.next = p;

}

* Thêm phẩn tử item vào cuối danh sách liên kết: void addLast(E item)

void addLast(E item)

{

//Tìm phần tử cuối

Node p = head;

While(p.getNext() != null)

P = p.getNext();

//gọi addafter

Node t = new Node(item);

p.setNext(t);

t.setNext(null);

}

* Xoá phần tử đầu tiên: void removeFirst()

void removeFirst()

{

head = head.getNext();

}

* Xoá phần tử sau sau phần tử hiện hành: void removeAfter(Node<E> curr)

void removeAfter(Node<E> curr)

{

//Curr.setNext(curr.getNext().getNext());

Node p = curr.getNext();

Curr.setNext(p.getNext());

}

Khó hơn: yêu cầu xoá phần tử có giá trị là item. -> tự làm

* Xoá phần tử cuối cùng: void removeLast()

void removeLast()

{

Node q, p = head;

While (p.getNext() != null)

{

q = p;

p = p.getNext();

}

q.setNext(null);

}

* Tìm kiếm phần tử item đầu tiên tồn tại trong DSLK. Nếu có thì trả về Node chứa item, nếu không có trả về null

Node search(E item)

{

Node p = head;

While(p != null)

{

If(p.getData().equals(item))

return p;

p = p.getNext();

}

return null;

}

1. Sử dụng sơ đồ lớp như câu 1 để cài đặt cấu trúc dữ liệu Stack và Queue. Viết các phương thức sau

* Thêm phần tử item vào Stack

Thực chất chính là hàm addFirst(E item)

* Lấy phần tử ra khỏi Stack

Thực chất là removeFirst()

* Thêm phần tử vào Queue

Thực chất addLast()

* Lấy phần tử ra khỏi Queue

Thực chất là removeFirst()

1. Sắp xếp:

* Cho biết ý tưởng của các thuật toán sắp xếp: Selection, Bubble, insertion, Merge sort, Quick sort, radix sort
* Cho mảng các số nguyên: 8 10 1 6 5 20 30 15. Chạy từng bước 6 thuật toán trên cho mảng này

Selecttion

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| 0 | 1 | 10 | 8 | 6 | 5 | 20 | 30 | 15 |
| 1 | 1 | 5 | 8 | 6 | 10 | 20 | 30 | 15 |
| 2 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 3 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 4 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 5 | 1 | 5 | 6 | 8 | 10 | 15 | 30 | 20 |
| 6 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |

Bubble

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| 0 | 1 | 8 | 10 | 5 | 6 | 15 | 20 | 30 |
| 1 | 1 | 5 | 8 | 10 | 6 | 15 | 20 | 30 |
| 2 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |
| 3 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |
| 4 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |
| 5 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |
| 6 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |

Insertion

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| 1 | 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| 2 | 1 | 8 | 10 | 6 | 5 | 20 | 30 | 15 |
| 3 | 1 | 6 | 8 | 10 | 5 | 20 | 30 | 15 |
| 4 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 5 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 6 | 1 | 5 | 6 | 8 | 10 | 20 | 30 | 15 |
| 7 | 1 | 5 | 6 | 8 | 10 | 15 | 20 | 30 |

Merge sort

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Mảng ban đầu | 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| Chia | 8 10 1 6 | | | | 5 20 30 15 | | | |
| 8 10 | | 1 6 | | 5 20 | | 30 15 | |
| 8 | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
| Gộp | 8 10 | | 1 6 | | 5 20 | | 15 30 | |
| 1 6 8 10 | | | | 5 15. 20 30 | | | |
| 1 5. 6. 8. 10. 15. 20. 30 | | | | | | | |

Quick sort:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 (pivot) | 10 | 1 | 6 | 5 | 20 | 30 | 15 |
|  | 1 | 6 | 5 | 10 | 20 | 30 | 15 |
| 5(pivot) | 1 | 6 | **8** | 10 (pivot) | 20 | 30 | 15 |
| 1 | **5** | 6 | **8** | **10** | 20 (pivot) | 30 | 15 |
| **1** | **5** | **6** | **8** | **10** | 15 | **20** | 30 |
| **1** | **5** | **6** | **8** | **10** | 15 | **20** | 30 |

Radix sort:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 08 | 10 | 01 | 06 | 05 | 20 | 30 | 15 |

(10,20,30) (01) (05, 15) (06) (08)

10 20 30 01 05 15 06 08

(01, 05, 06, 08) (10, 15) (20) (30)

1 5 6 8 10 15 20 30

* Đánh giá độ phức tạp cho 6 thuật toán trên.

1. Hashing:

* Cho hash method: hash(k) = k%m. Cho hash table có kích thước m = 7.
* Lần lượt thêm các phần tử sau vào hash table: 8 10 1 6 5 20 15. Nếu xảy ra đụng độ hãy sử dụng phương pháp Linear Probing để giải quyết đụng độ.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Thêm 8 |  | 8 |  |  |  |  |  |
| Thêm 10 |  | 8 |  | 10 |  |  |  |
| Thêm 1 |  | 8 | 1 | 10 |  |  |  |
| Thêm 6 |  | 8 | 1 | 10 |  |  | 6 |
| Thêm 5 |  | 8 | 1 | 10 |  | 5 | 6 |
| Thêm 20 | 20 | 8 | 1 | 10 |  | 5 | 6 |
| Thêm 15 | 20 | 8 | 1 | 10 | 15 | 5 | 6 |

* Cho biết có bao nhiêu phép so sánh khi tìm kiếm phần tử 15 trên hash table.

+ Đầu tiên 15%7 = 1

+ Sau đó tìm từ vị trí 1 -> không tìm thấy -> tìm vị trí 2 -> không tìm thấy -> tìm vị trí 3 -> ko tìm thấy -> tìm vị trí 4 -> tìm thấy

+ có tổng cộng 4 phép so sánh khi tìm phần tử 15 trong hash table.

* Lần lượt xoá các phần tử 1, 6, 15. Thêm phần tử 22, 27, 29.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 20 | 8 | 1 | 10 | 15 | 5 | 6 |
| Xoá 1 | 20 | 8 | x | 10 | 15 | 5 | 6 |
| Xoá 6 | 20 | 8 | x | 10 | 15 | 5 | x |
| Xoá 15 | 20 | 8 | x | 10 | x | 5 | x |
| Thêm 22 | 20 | 8 | 22 | 10 | x | 5 | x |
| Thêm 27 | 20 | 8 | 22 | 10 | x | 5 | 27 |
| Thêm 29 | 20 | 8 | 22 | 10 | 29 | 5 | 27 |
|  |  |  |  |  |  |  |  |

Ghi chú: x là đại diện cho vị trí bị xoá

* Cho biết có bao nhiêu phép so sánh khi tìm kiếm phần tử 27 trên hash table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Thêm 29 | 20 | 8 | 22 | 10 | 29 | 5 | 27 |
|  |  |  |  |  |  |  |  |

+ đầu tiên: 27 % 7 = 6

+ Tìm vị trí số 6 -> tìm thấy

Vậy có 1 phép so sánh khi tìm kiếm phần tử 27 trên hash table.

**Phần 2: cây**

1. Cây BST: Cho dãy các khoá: 20 30 15 10 40 5 22 35 2 45

* Dựng cây BST với các khoá trên
* Chạy từng bước duyệt cây BST với các khoá trên theo các thuật toán: LNR, LRN, NLR, RNL, RLN, NRL
* Tìm predecessor, successor cho các khoá: 2, 45, 20.
* Viết thuật toán cho các bài toán sau: tìm max, min, tính tổng cây BST; tính tổng cây con bên trái, bên phải của Node x.
* Xoá các node: 20, 10, 5.

1. Cây AVL: Cho dãy các khoá: 20 30 40 15 25 23 5 22 35 2

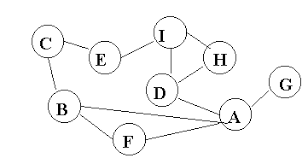
* Dựng cây AVL với các khoá trên
* Chạy từng bước duyệt cây AVL với các khoá trên theo các thuật toán: LNR, LRN, NLR, RNL, RLN, NRL
* Tìm predecessor, successor cho các khoá: 2, 45, 20.
* Lần lượt xoá các node: 2,5,15.

1. Cây Binary Max Heap: Cho dãy các khoá: 20 30 40 15 25 23 5 22 35 2

* Dựng cây Binary max heap
* Thực hiện dựng lại cây khi lấy các phần tử max ra khỏi cây.
* Thực hiện từng bước sắp tăng dần dựa trên cây binary max heap.

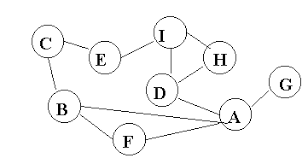
**Phần 3: Đồ thị**

1. Duyệt đồ thị bằng thuật toán BFS, DFS: (ưu tiên chọn đỉnh kế tiếp theo tứ tự alphabet, hoặc thứ tự số tự nhiên)
   1. Đỉnh gốc là: A



BFS

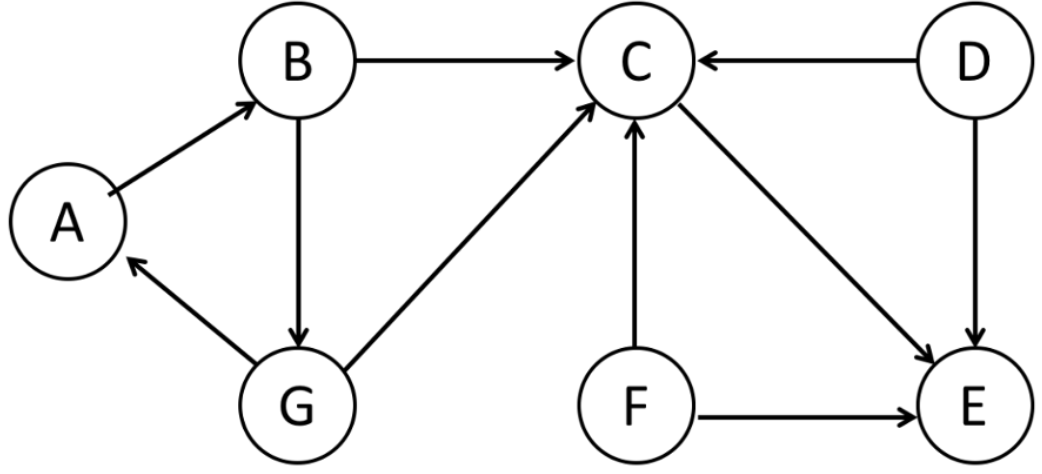
|  |  |
| --- | --- |
| Queue | Output |
| A | A |
| BDFG | ABDFG |
| DFGC | ABDFGC |
| FGCIH | ABDFGCIH |
| GCIH |  |
| CIH |  |
| IHE | ABDFGCIHE |
| HE |  |
| E |  |
| - | ABDFGCIHE |



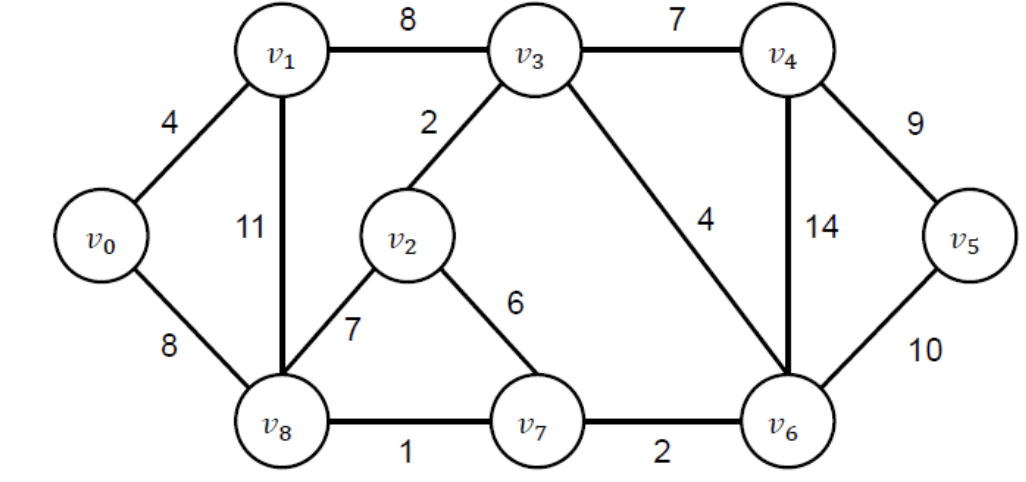
DFS

|  |  |
| --- | --- |
| Stack | Output |
| ABCEIDH | ABCEIDH |
| ABCEID |  |
| ABCEI |  |
| ABCE |  |
| ABC |  |
| ABF | ABCEIDHF |
| AB |  |
| AG | ABCEIDHFG |
| A |  |
| - | ABCEIDHFG |

* 1. Đỉnh gốc là: A



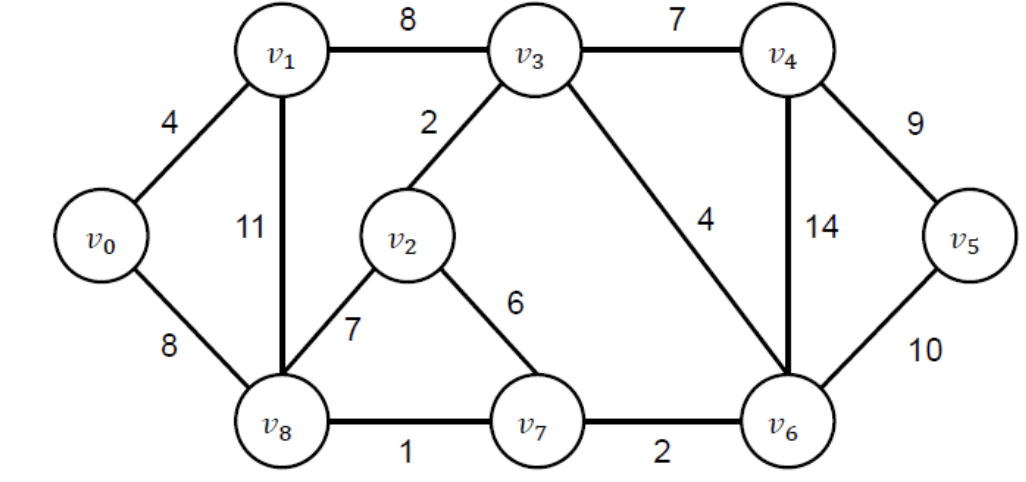
1. Tìm cây khung nhỏ nhất bằng thuật toán Prim và Kruskal cho các đồ thị sau:
   1. Đỉnh gốc là: v0 (Prim)



Prim

|  |  |  |
| --- | --- | --- |
| Tập hợp đỉnh đã chọn | Cạnh được chọn | MST |
| {0} | 0-1 | 9  7  2  4  2  1  8  4 |
| {0,1} | 0-8 |
| {0,1,8} | 8-7 |
| {0,1,8,7} | 7-6 |
| {0,1,8,7,6} | 6-3 |
| {0,1,8,7,6,3} | 3-2 |
| {0,1,8,7,6,3,2} | 3-4 |
| {0,1,8,7,6,3,2,4} | 4-5 |
| {0,1,8,7,6,3,2,4,5} |  |

Đã phủ đủ các đỉnh, trọng số của cây MST là: 4+8+1+2+4+2+7+9 = 37

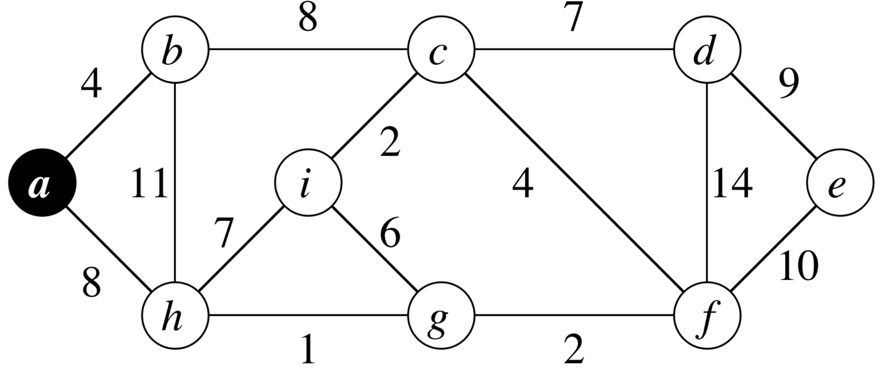


Kruskal

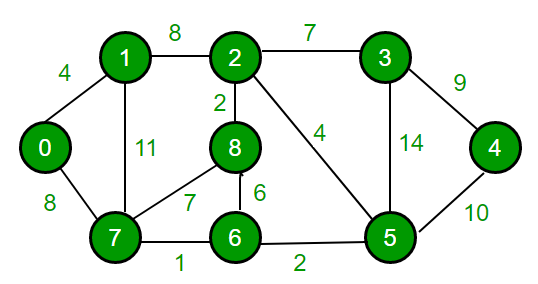
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| u | v | Trọng số | Chọn | MST |
| 7 | 8 | 1 | OK | 8  9  7  2  4  2  1  4 |
| 2 | 3 | 2 | OK |
| 6 | 7 | 2 | OK |
| 0 | 1 | 4 | OK |
| 3 | 6 | 4 | OK |
| 2 | 7 | 6 | CHU TRÌNH |
| 2 | 8 | 7 | CHU TRÌNH |
| 3 | 4 | 7 | OK |
| 1 | 3 | 8 | OK |
| 0 | 8 | 8 | CHU TRÌNH |
| 4 | 5 | 9 | OK |
| 5 | 6 | 10 | CHU TRÌNH |
| 1 | 8 | 11 | CHU TRÌNH |
| 4 | 6 | 14 | CHU TRÌNH |

Trọng số của cây MST là: 1+2+2+4+4+7+8+9 = 37

* 1. Đỉnh gốc là: A (Prim)



1. Sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất cho các đồ thị sau:
   1. Đỉnh gốc là 0



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Khởi tạo | -,0 | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ |
| 1 | - | 0,4\* | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ | 0,8 | -,∞ |
| 2 | - | - | 1,12 | -,∞ | -,∞ | -,∞ | -,∞ | 0,8\* | -,∞ |
| 3 | - | - | 1,12 | -,∞ | -,∞ | -,∞ | 7,9\* | - | 7,15 |
| 4 | - | - | 1,12 | -,∞ | -,∞ | 6,11\* | - | - | 7,15 |
| 5 | - | - | 1,12\* | 5,25 | 5,21 | - | - | - | 7,15 |
| 6 | - | - | - | 2,19 | 5,21 | - | - | - | 2,14\* |
| 7 | - | - | - | 2,19\* | 5,21 | - | - | - | - |
| 8 | - | - | - | - | 5,21\* | - | - | - | - |

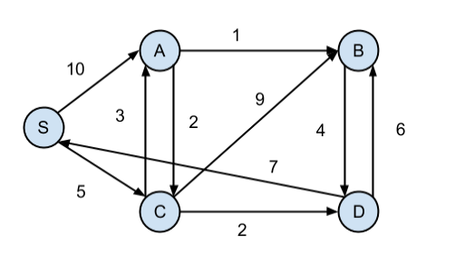
Vậy đường đi ngắn nhất từ đỉnh 0 đến các đỉnh còn lại: là

0 – 1 – 2 – 3

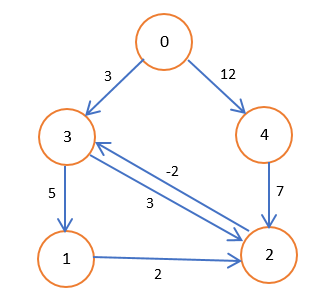
0 – 1 – 2 – 8

0 – 7 – 6 – 5 – 4

* 1. Đỉnh gốc là S



1. Sử dụng thuật toán Bellman Ford để tìm đường đi ngắn nhất cho các đồ thị sau:
   1. Đỉnh gốc là 0



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| Khởi tạo | -,0 | -,∞ | -,∞ | -,∞ | -,∞ |
| 1 | - | -,∞ | -,∞ | 0,3 | 0,12 |
| 2 | - | 3,8 | 3,6 | 0,3 | 0,12 |
| 3 | - | 3,8 | 3,6 | 0,3 | 0,12 |
| 4 | - |  |  |  |  |
| 5 | - |  |  |  |  |

Thuật toán hội tụ -> dừng

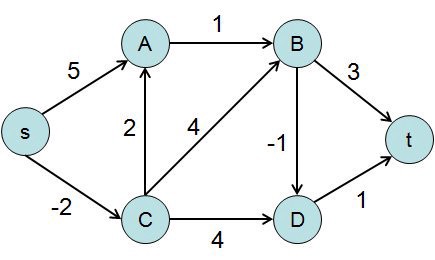
Đường đi ngắn nhất từ đỉnh 0 đến các đỉnh còn lại:

0 – 3 – 1

0 – 3 – 2

0 – 4

* 1. Đỉnh gốc là S



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | s | A | B | C | D | t |
| Khởi tạo | -,0 | -,∞ | -,∞ | -,∞ | -,∞ | -,∞ |
| 1 | - | S,5 | -,∞ | S,-2 | -,∞ | -,∞ |
| 2 | - | C,0 | C,2 | S,-2 | C,2 | -,∞ |
| 3 | - | C,0 | A,1 | S,-2 | B,1 | D,3 |
| 4 | - | C,0 | A,1 | S,-2 | B,0 | D,2 |
| 5 | - | C,0 | A,1 | S,-2 | B,0 | D,1 |
| 6 | - | C,0 | A,1 | S,-2 | B,0 | D,1 |

Thuật toán hội tụ.

Đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại là:

S – C – A – B – D - T