TRƯỜNG ĐẠI HỌC XÂY DỰNG
KHOA CÔNG NGHỆ THÔNG TIN
------------o0o------------

BÁO CÁO BÀI TẬP
PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN CHƯƠNG 2

ĐỀ TÀI:

Phân tích các thuật toán bằng phương pháp chia để trị và so sánh với các thuật toán không đệ quy

GV hướng dẫn: Thầy Phạm Hồng Phong

Nhóm sinh viên thực hiện (Nhóm 4):
Dương Công Sơn 64CS3 MSSV 167464
Phạm Hoàng Đăng Trung 64CS3 MSSV 1551664
Trịnh Ngọc Dương 64CS3 MSSV 39864
Hoàng Thị Thanh Tú 64CS3 MSSV 208764
Bùi Quốc Việt 64CS3 MSSV 222964

Hà Nội, tháng 7 năm 2021.

I. Lý thuyết

4.Trình bày thuật toán Prim

4.1. Bài toán

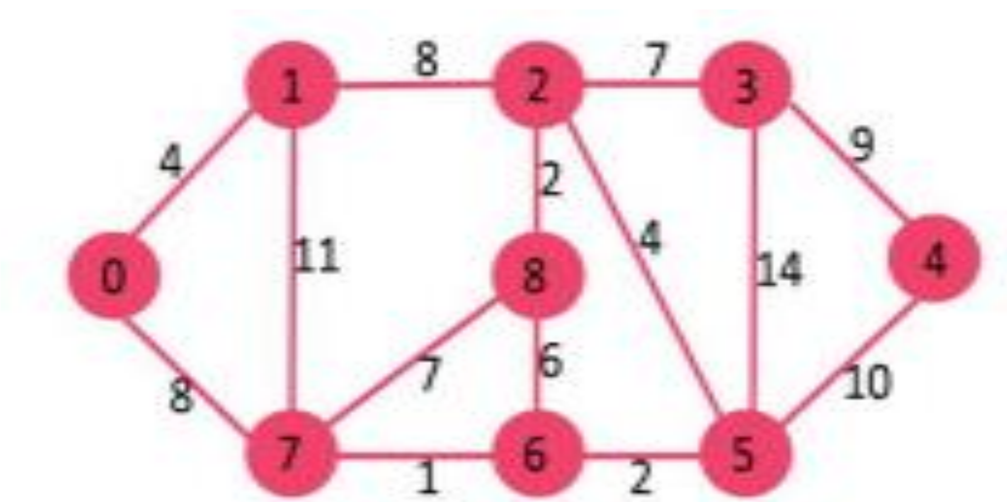
Bài toán cây khung (cây bao trùm) nhỏ nhất của đồ thị là một trong số những bài toán tối ưu trên đồ thị có nhiều ứng dụng khác nhau trong đời sống.

Giả sử ta có một đồ thị vô hướng, liên thông và có trọng số G =(V,E). Cây bao trùm tối thiểu của đồ thị G là một cây T có tập các nút là V là E, tập các cạnh và tổng số độ dài các cạnh trong T là nhỏ nhất.

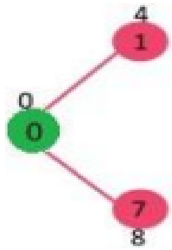
4.2 Ý tưởng thuật toán Prim

Nạp dần các đỉnh vào cây khung. Mỗi lần chọn một đỉnh chưa nạp sao cho đỉnh đó kề và gần nhất với các đỉnh đã nạp.

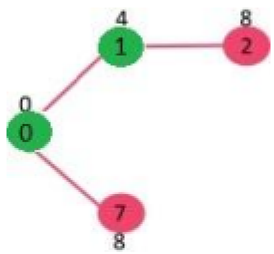
4.3 Ví dụ



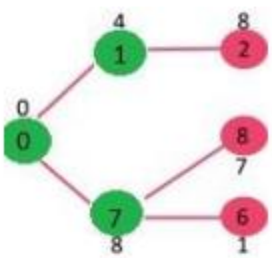
Bước 1:



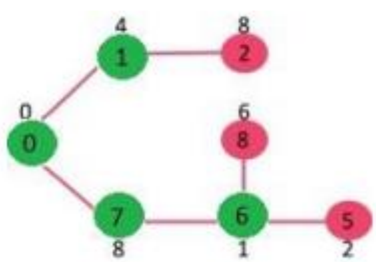
Bước 2:



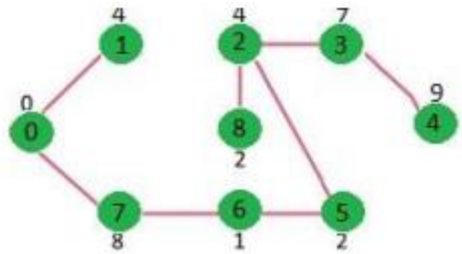
Bước 3:



Bước 4:



Cây sau khi hoàn thành:



4.4. Mô tả thuật toán

- Dữ liệu vào: Một đồ thị có trọng số liên thông với tập hợp đỉnh *V* và tập hợp cạnh *E* (trọng số có thể âm). Đồng thời cũng dùng *V* và *E* để ký hiệu số đỉnh và số cạnh của đồ thị.
- Khởi tạo: $V_{mới}$ = {*x*}, trong đó *x* là một đỉnh bất kì (đỉnh bắt đầu) trong *V*, $E_{mới}$ = {}
- Lặp lại cho tới khi $V_{mới}$ = *V*:
  - Chọn cạnh (*u*, *v*) có trọng số nhỏ nhất thỏa mãn *u* thuộc $V_{mới}$ và *v* không thuộc $V_{mới}$ (nếu có nhiều cạnh như vậy thì chọn một cạnh bất kì trong chúng)
  - Thêm *v* vào $V_{mới}$, và thêm cạnh (*u*, *v*) vào $E_{mới}$
- Dữ liệu ra: $V_{mới}$ và $E_{mới}$ là tập hợp đỉnh và tập hợp cạnh của một cây bao trùm nhỏ nhất

4.5. Chứng minh tính đúng

Chứng minh: Mỗi cây con T_i, i = 0, …, n-1, sinh ra bằng thuật toán Prim là một cây bao trùm tối thiểu

Hiển nhiên: T_0 là một phần của cây bao trùm tối thiểu

Ta giả sử: T_{i-1} là một phần của cây bao trùm tối thiểu. Ta chứng minh T_i cũng là một cây bao trùm tối thiểu

Sử dụng phương pháp Phản chứng, ta có:

Giả sử, cây bao trùm tối thiểu T của G không chứa T_i, tức là có cạnh e = (v,u) là cạnh có trọng số nhỏ nhất nối một đỉnh thuộc T_{i-1} với 1 đỉnh không thuộc T_{i-1}, và e không thuộc cây tối thiểu T

- Nếu thêm e vào T sẽ tạo ra một chu trình: chu trình chứa cạnh e′ = (v′, u′) nối một đỉnh v′ ∈ T_{i-1} với một đỉnh u′∉T_{i-1}. Nếu xóa đi cạnh e′ của chu trình này sẽ nhận được một cây bao trùm khác chứa e có trọng số nhỏ hơn T.
- Nếu thêm e vào T sẽ tạo ra một chu trình: chu trình chứa cạnh e′ = (v′, u′) nối một đỉnh v′ ∈ T_{i-1} với một đỉnh u′∉T_{i-1}. Nếu xóa đi cạnh e′ của chu trình này sẽ nhận được một cây bao trùm khác chứa e có trọng số nhỏ hơn T.

Như vậy trái với giả thiết cây bao trùm tối thiểu không chứa T_i =>đpcm

4.6. Độ phức tạp thuật toán

*Sử dụng ma trận kề biểu diễn đồ thị:

Code	Thời gian

Prim(G) ≡
// Input: G = (V, E)
// Output: ET , tập các cạnh của cây bao trùm tối thiểu của G
VT = {v0}
ET = ∅
	O(2)

for i=1..|V| - 1 tìm cạnh có trọng số nhỏ nhất e∗=(v∗,u∗) trong tất cả các cạnh (v, u) mà v ∊ VT và u ∊ V - VT; ∪ {u∗} ∪ {e*} VT = VT ET = ET **endf;** return ET; **End**	O(V^2)

Từ trên ta có độ phức tạp khi sử dụng ma trận kề biểu diễn đồ thị:

=> T(n) = O(V^2)

*Sử dụng cây nhị phân đống nhỏ nhất (min heap) và danh sách kề biểu diễn đồ thị:

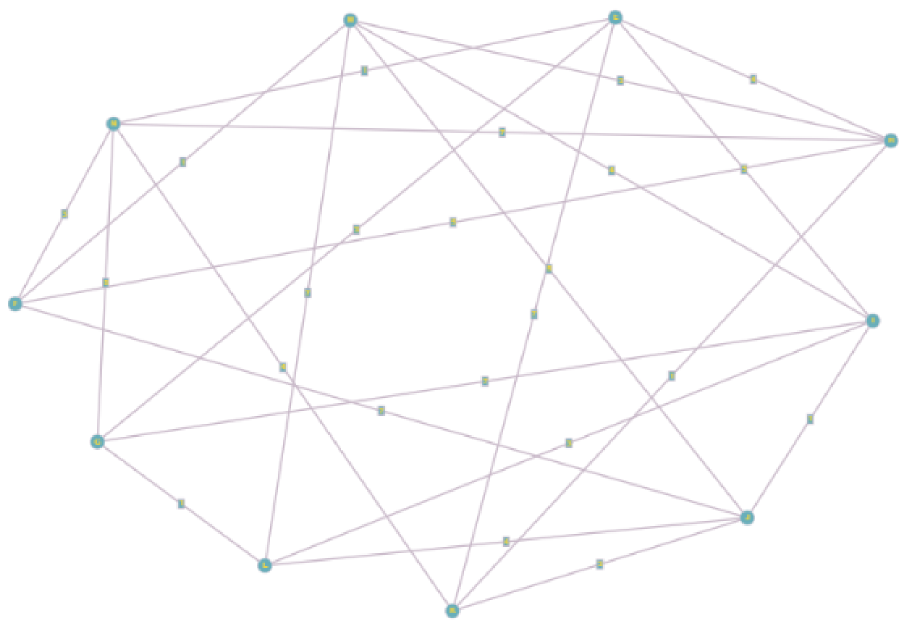
=> T(n) = O(E log V)

-> Thời gian chạy giảm xuống

4.7. Code và Thực nghiệm

Thực nghiệm:

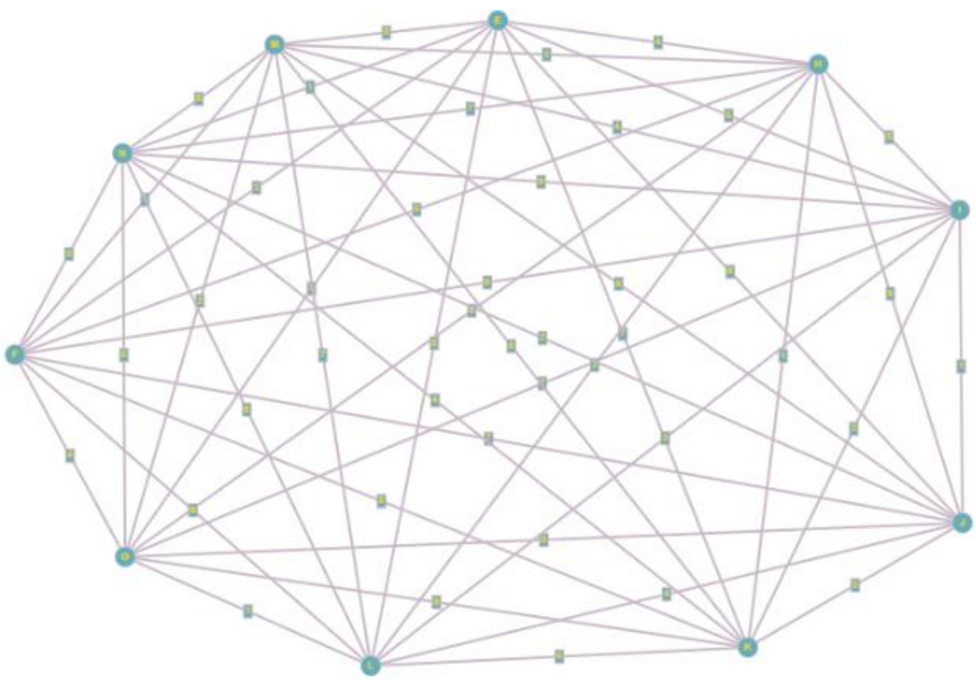
Với đồ thị có 10 đỉnh và 2x cạnh.



Thuật toán Prim’s chạy mất: 8.846 milliseconds

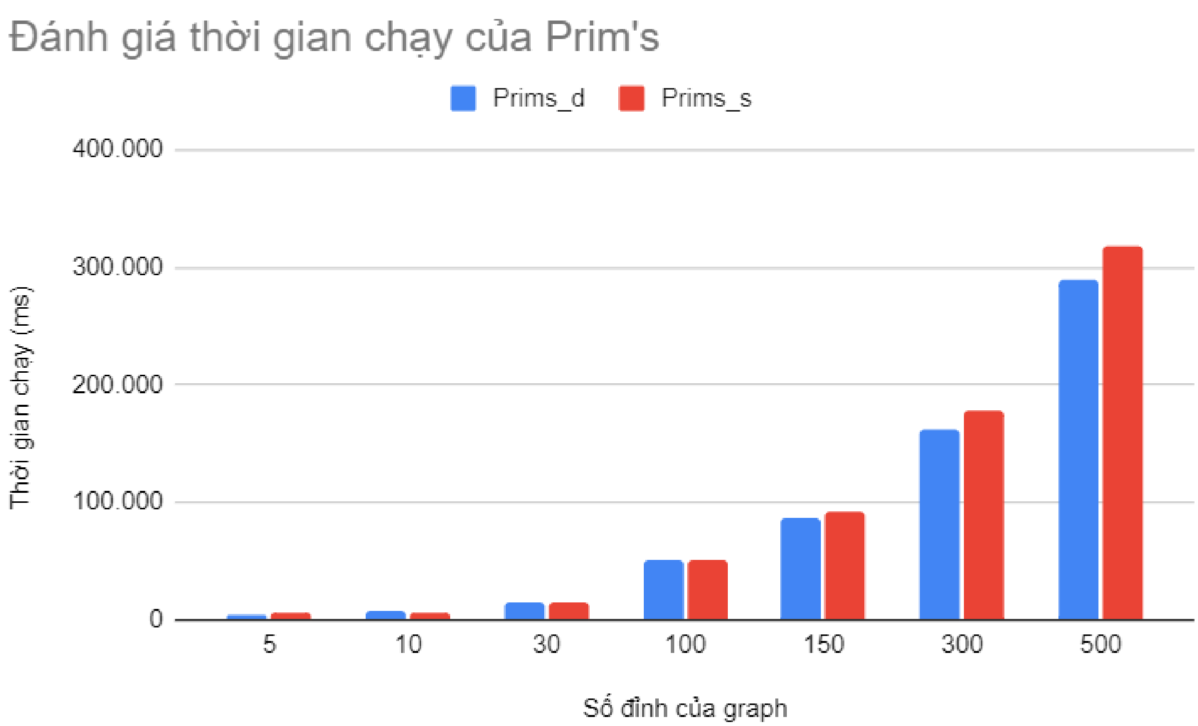
Thực nghiệm:

Với đồ thị có 10 đỉnh và tất cả cách đỉnh đều có thể đi trực tiếp tới nhau.



Thuật toán Prim’s chạy mất: 8.4206 milliseconds

Thực nghiệm: So sánh giữa độ thị dày đặc và đồ thị thưa thớt



5. Trình bày thuật toán Kruskal

5.1. Bài toán

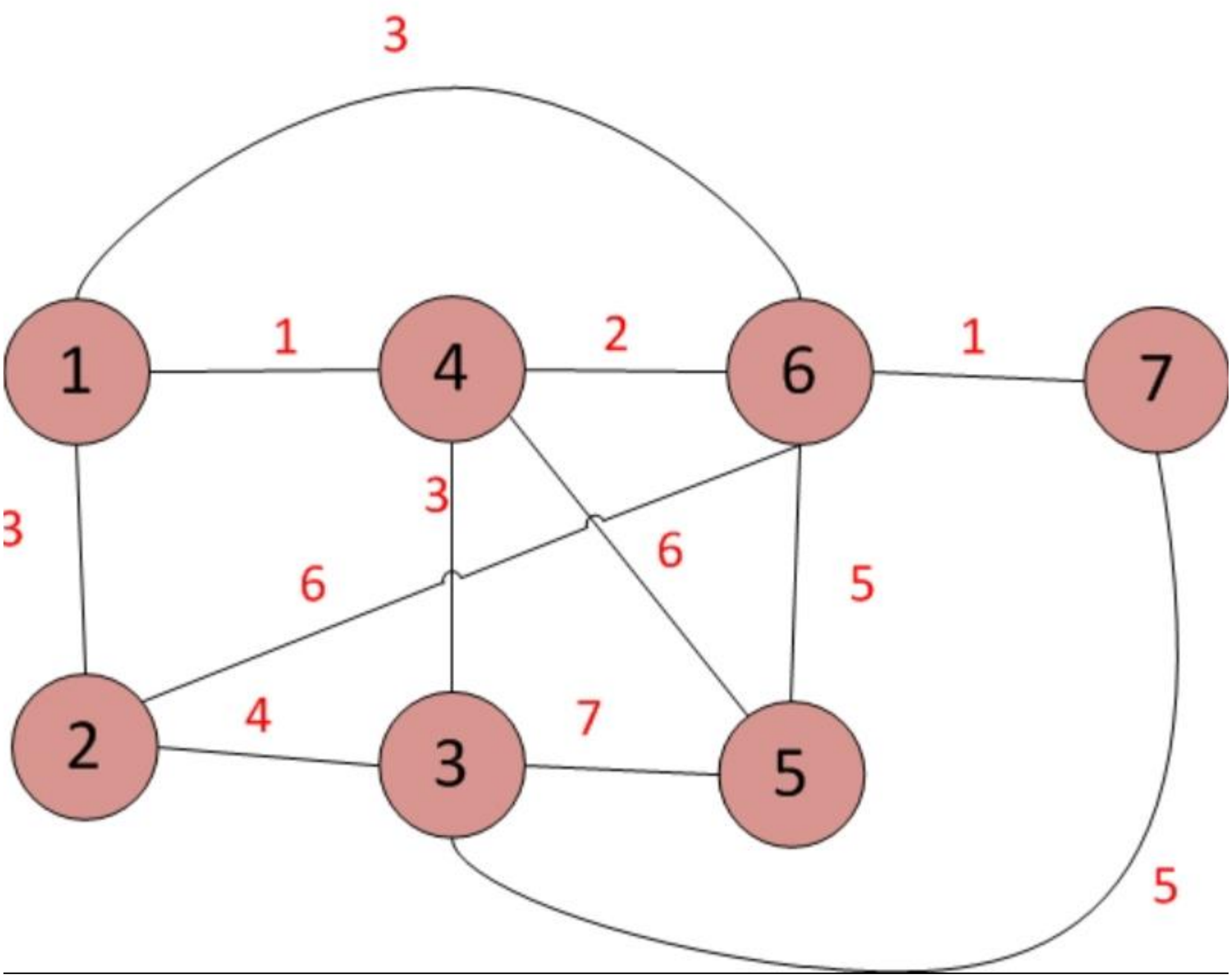
Giả sử ta có một đồ thị vô hướng, có trọng số, liên thông G = (V, E). Cây bao trùm tối thiểu (Minimum Spanning Tree – MST) của đồ thị G là một cây T có tập các nút là V, tập các cạnh là E và tổng độ dài các cạnh trong T là nhỏ nhất.

5.2. Ý tưởng thuật toán Kruskal

Ban đầu mỗi đỉnh là một cây riêng biệt, ta tìm cây khung nhỏ nhất bằng cách duyệt các cạnh theo trọng số từ nhỏ đến lớn, rồi hợp nhất các cây lại với nhau

5.3. Ví dụ

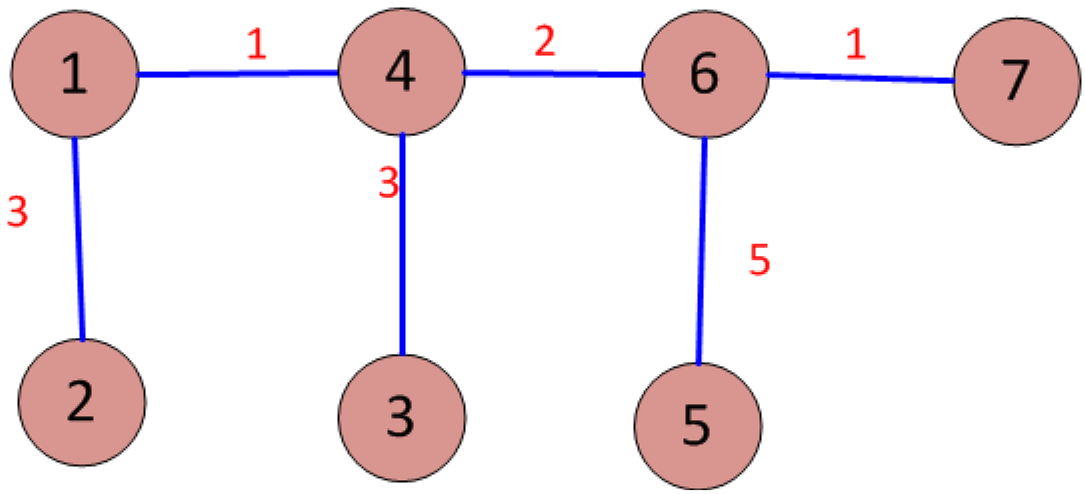
Cho đồ thị vô hướng như sau:



Sắp xếp các cạnh theo trọng số tăng dần

Điểm đầu	Điểm cuối	Trọng số
1	4	1
6	7	1
4	6	2
1	2	3
1	6	3
3	4	3
2	3	4
3	7	5
5	6	5
2	6	6
4	5	6
3	5	7

Sau khi thêm các cạnh vào ta được đồ thị :



Tổng trọng số: 3 + 1 + 3 + 2 + 5 + 1 = 15

5.4. Mô tả thuật toán

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị *G*. Thuật toán bao gồm các bước sau.

- Khởi tạo *F* (tập hợp các cây), trong đó mỗi đỉnh của *G* tạo thành một cây riêng biệt
- Khởi tạo tập *S* chứa tất cả các cạnh của *G*
- Chừng nào *S* còn khác rỗng và *F* gồm hơn một cây
  - Xóa cạnh nhỏ nhất trong *S*
  - Nếu cạnh đó nối hai cây khác nhau trong *F*, thì thêm nó vào *F* và hợp hai cây kề với nó làm một
  - Nếu không thì loại bỏ cạnh đó.

Khi thuật toán kết thúc, F chỉ gồm đúng một cây và đó là một cây bao trùm nhỏ nhất của đồ thị *G*.

5.5. Chứng minh tính đúng

Chứng minh gồm hai phần: chứng minh kết quả thuật toán là một cây bao trùm và cây bao trùm đó là nhỏ nhất.

Phần 1: Chứng minh kết quả thuật toán là một cây bao trùm

F luôn là một rừng do việc nối hai cây bằng một cạnh luôn tạo ra một cây mới. Giả thiết phản chứng *F* gồm ít nhất hai cây *A* và *B*. Khi cạnh đầu tiên nối các đỉnh trong *A* của *F* với phần còn lại của đồ thị được xem xét (cạnh này tồn tại do *G* liên thông) thì rõ ràng thuật toán sẽ chọn nó. Vì vậy *A* không thể là một cây trong *F* khi thuật toán kết thúc. Do đó, *F* liên thông và là một cây bao trùm.

Phần 2: Cây bao trùm đó là nhỏ nhất.

Ta chứng minh mệnh đề P sau đây bằng quy nạp: Nếu *F* là tập hợp các cạnh đã chọn tại bất kì thời điểm nào trong quá trình thực thi thuật toán thì tồn tại cây bao trùm nhỏ nhất chứa *F*.

Hiển nhiên: P đúng khi thuật toán bắt đầu vì *F* là rỗng.

Ta giả sử P là đúng cho một tập hợp *F*

Chứng minh bằng phương pháp Quy nạp:

Giả sử *T* là một cây bao trùm nhỏ nhất chứa *F*. Nếu cạnh được thêm vào tiếp theo là *e* cũng nằm trong *T*, thì P đúng cho $F + e$. Nếu không, thì $T + e$ chứa chu trình *C* và tồn tại cạnh *f* nằm trên *C* nhưng không trong *F*. (Nếu không có cạnh *f*, thì không thể thêm *e* vào *F*, do sẽ tạo ra chu trình *C* trong *F*.)

Do đó $T - f + e$ là một cây, và nó có cùng trọng số với *T*, do *T* có trọng số nhỏ nhất và *f* không thể nhỏ hơn *e*, vì nếu không thuật toán đã xem xét *f* trước *e* và chọn *f*. Vì vậy $T - f + e$ là một cây bao trùm nhỏ nhất chứa $F + e$ và P là đúng.

Như vậy, P đúng khi thuật toán kết thúc và *F* là một cây bao trùm. Điều này chỉ có thể xảy ra nếu *F* là một cây bao trùm nhỏ nhất.

5.6 Độ phức tạp thuật toán

Mã giả	Thời gian

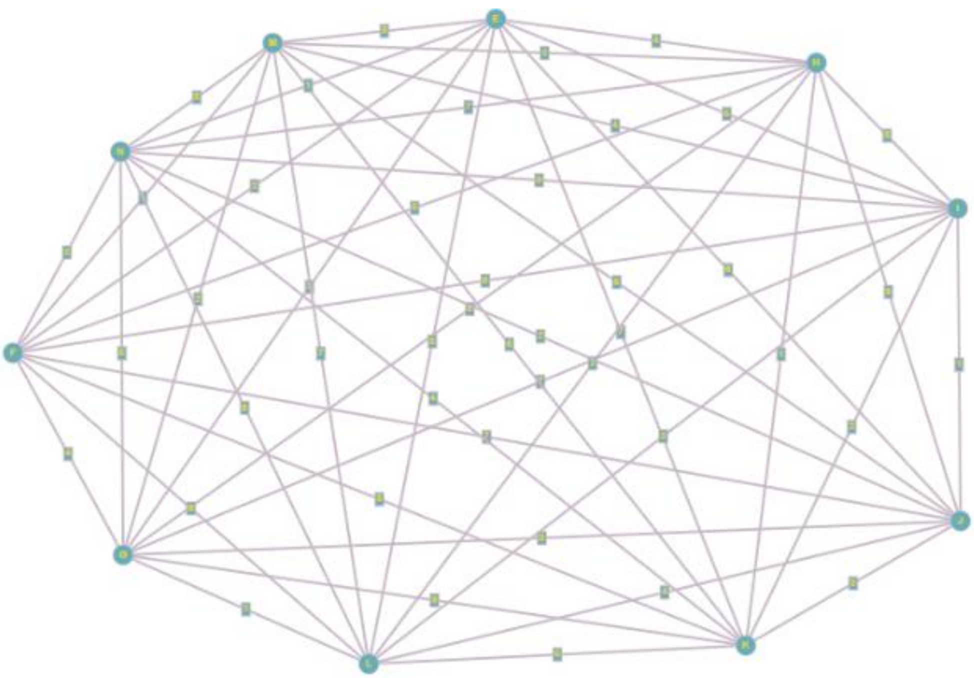
Kruskal(G):
A=∅
For each đỉnh v ∈ G.V:
    Make_Set(v)
For each cạnh(u,v) ∈ G.E sắp xếp theo thứ tự tăng dần theo trọng số (u, v):
    If Find_Set(u) ≠ Find_Set(v):
    A= A∪{(u,v)}
    Union(u,v)
Return A
	
	T(1)
	T(V)
	
	T(ElogE)
	T(ElogV)

Độ phức tạp thuật toán: T(n) = O((V+E)logV) = O(ElogV)

5.7. Code và Thực nghiệm

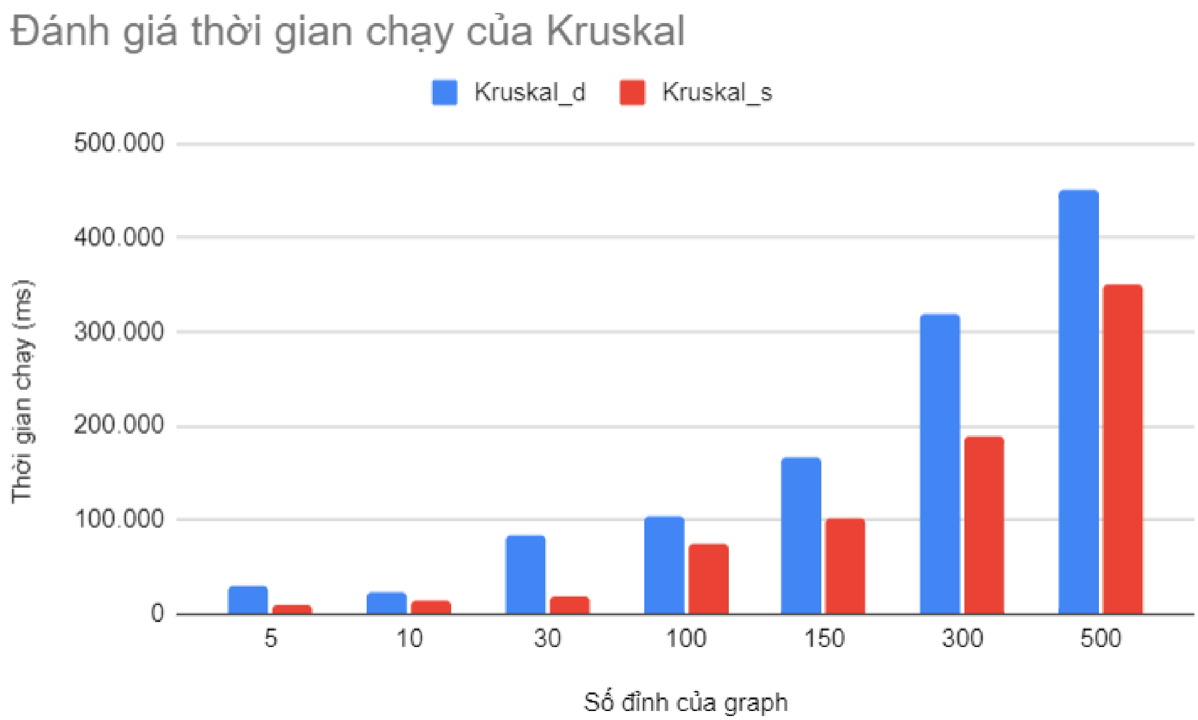
Thực nghiệm:

Với đồ thị có 10 đỉnh và tất cả cách đỉnh đều có thể đi trực tiếp tới nhau.



Thuật toán Kruskal chạy mất: 24.3807 milliseconds

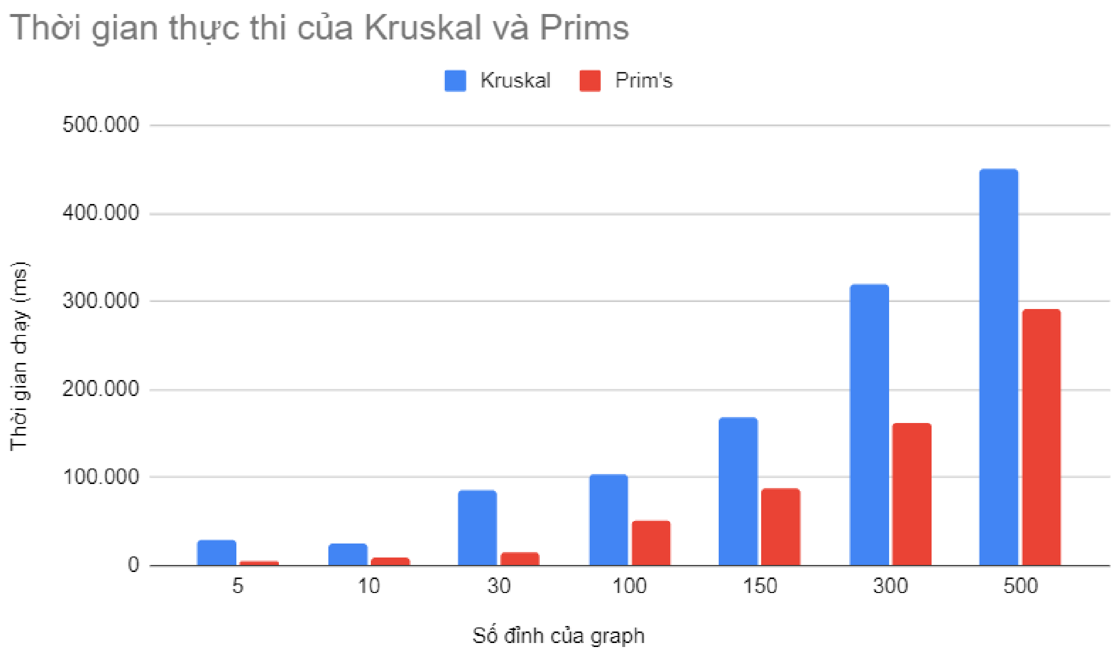
Thực nghiệm: So sánh giữa độ thị dày đặc và đồ thị thưa thớt.



Đánh giá thời gian chạy của Kruskal
Kruskal_d
Kruskal_s
500.000
400.000
300.000
200.000
100.000
0
Thời gian chạy (ms)
5
10
30
100
150
300
500
Số đỉnh của graph

II. So sánh và đánh giá

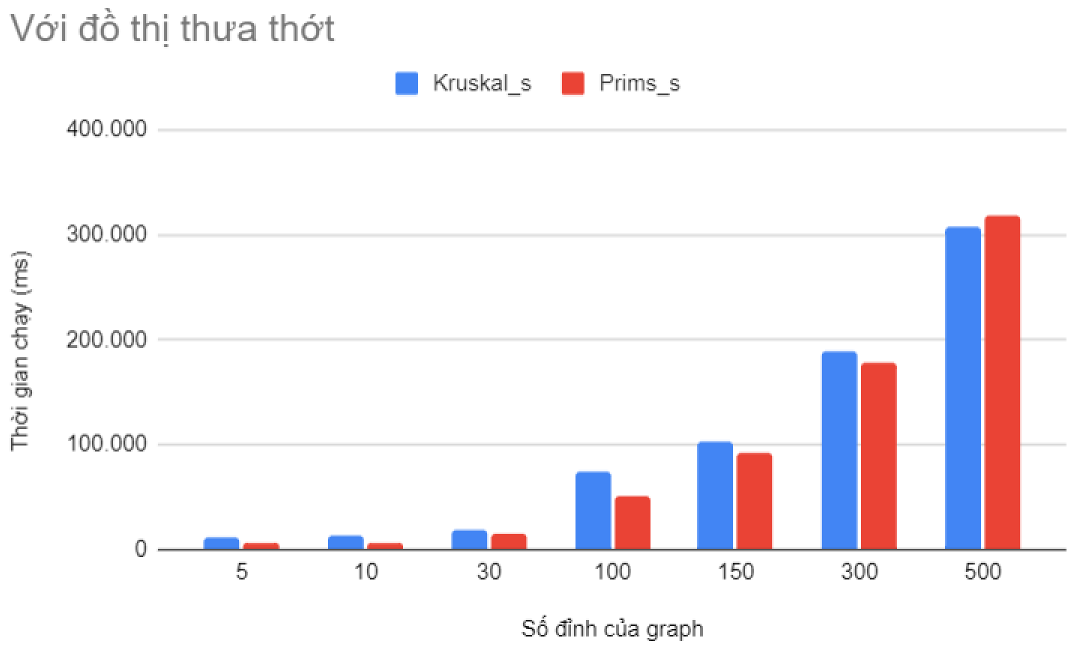
1. Với đồ thị dạng dày.



Dense	5	10	30	100	150	300	500
Kruskal	29.115	23.306	84.273	103.662	167.412	318.399	451.386
Prims	4.781	7.763	14.906	51.098	86.096	161.767	290.121

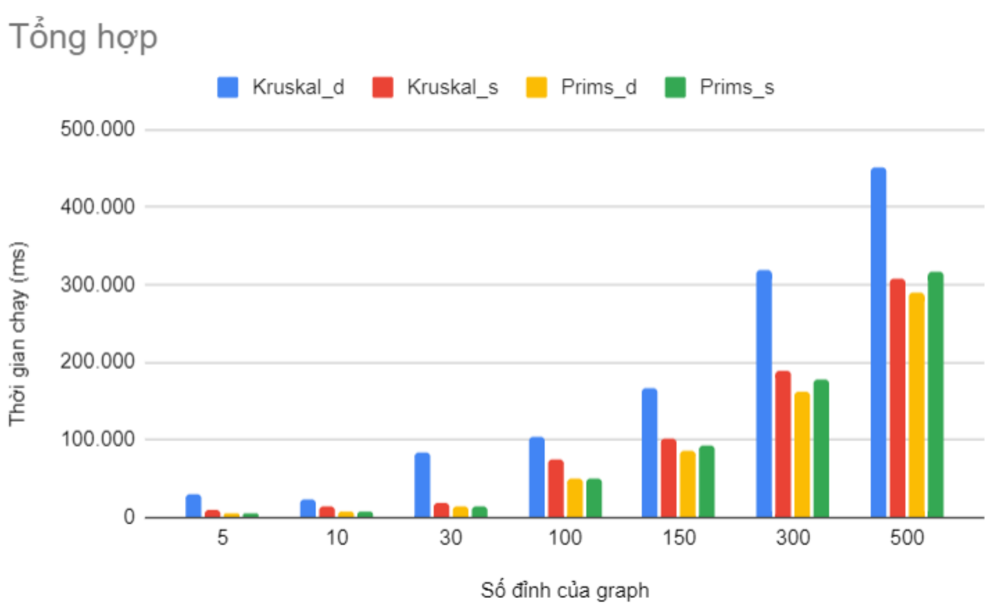
Khi đồ thị ở dạng dày ta thấy rõ thuật toán Prim’s chạy nhanh hơn thuật toán Kruskal.

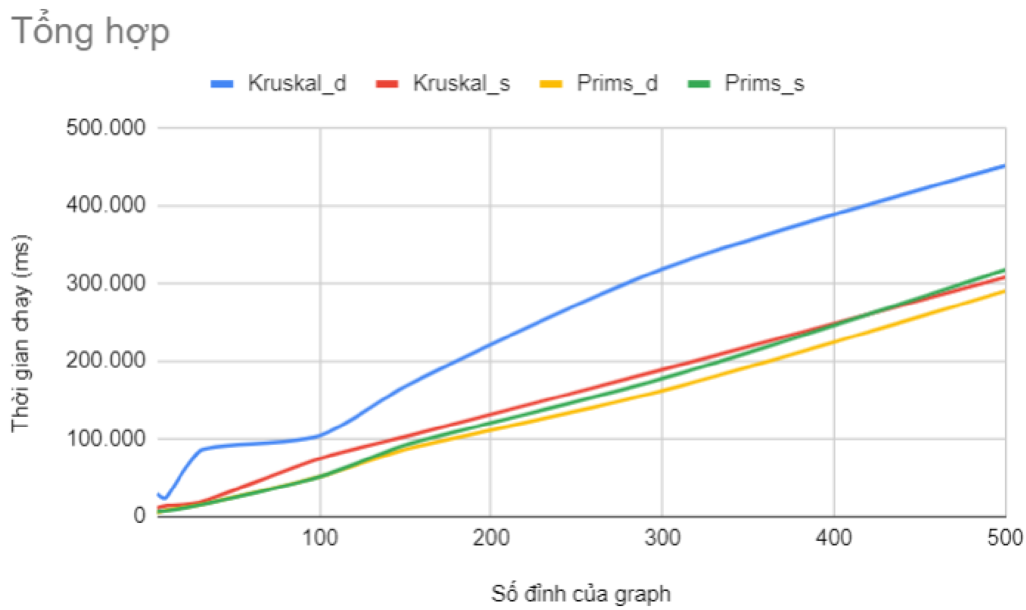
2. Với đồ thị dạng thưa thớt.



Sparse	5	10	30	100	150	300	500
Kruskal	10.909	13.514	17.968	74.318	102.644	188.836	307.805
Prims	6.462	7.036	14.655	51.044	91.818	177.365	317.414

3. Tổng hợp.





Nhận xét.

- Khi đồ thị ở dạng dày thì thuật toán Prim's chạy nhanh hơn so với Krusal.
- Còn khi ở đồ thị dạng thưa thới, có vẻ thời gian thực thi của hai thuật toán là ngang nhau.
- Ta thấy rõ rằng thuật toán Kruskal thì thực hiện tốt hơn khi đồ thị có ít cạnh.

III. Tổng kết.

Thông qua việc tìm hiểu nghiên cứu đã giúp nhóm chúng em có cái nhìn toàn cảnh hơn trong việc thực hiện sử dụng giải thuật tham lam áp dụng vào từng bài toán được đặt ra. Từ các thuật toán ở trên có thể thấy được giải thuật tham lam giải quyết các vấn đề tương đối nhanh, có tính ứng dụng cao trong việc xử lý các bài toán thực tế. Trong quá trình thực hiện đề tài không tránh khỏi những sai sót, nhóm chúng em rất mong nhận được sự góp ý của thầy, qua đó đánh giá giúp nhóm hoàn thiện bài tập