ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO LAB 1

BỘ MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

NGƯỜI THỰC HIỆN
Họ và tên: Đỗ Đạt Thành
MSSV: 20127411
Lớp: 20CLC04

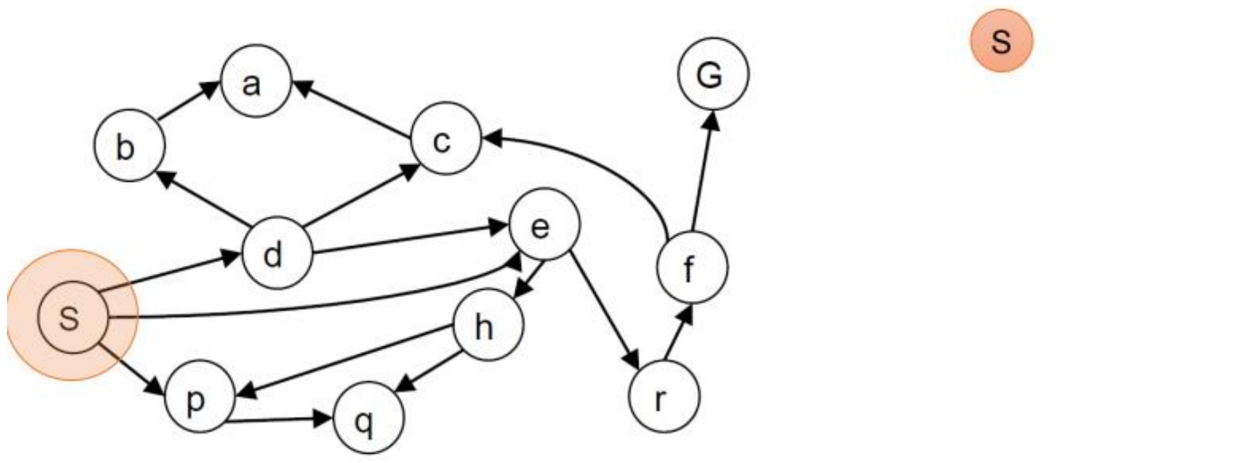
TP. HỒ CHÍ MINH – NĂM 202

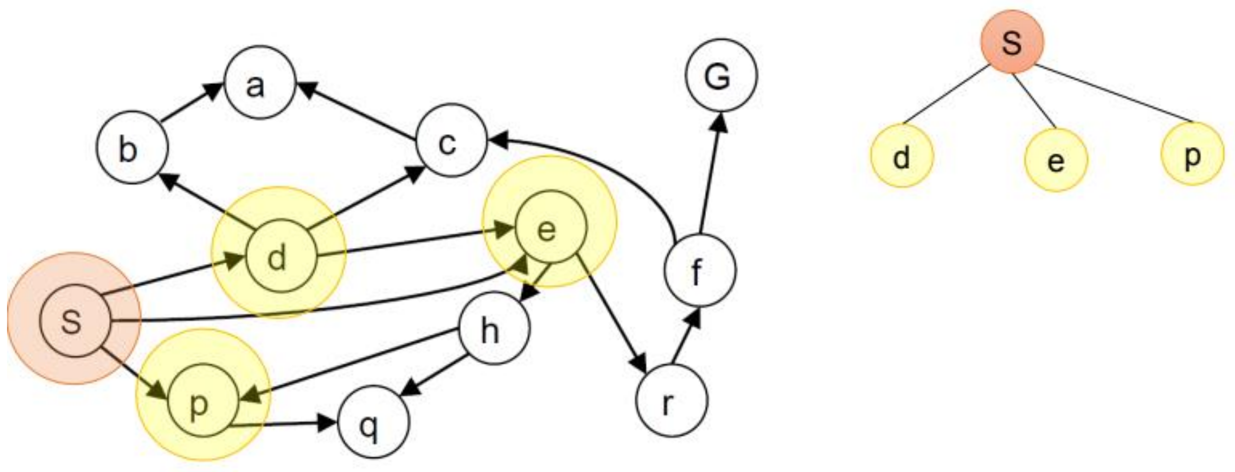
I. Breadth-first search

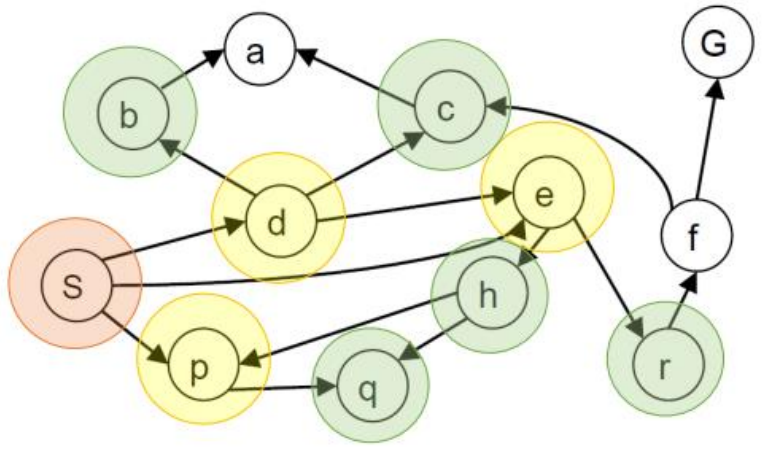
a. Idea

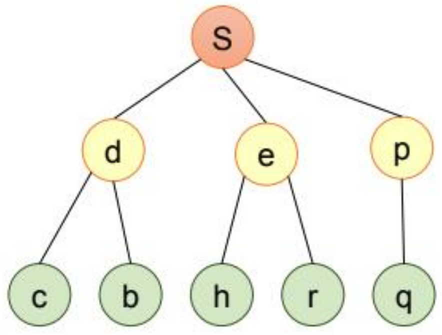
- Create place contain next edge can go and visited
- Perform loop to check edge if it is possible to go, draw it and put it into visited array
- Robot just moves 4 directions so just check 4 times allow 4 directions
- After found the goal, show the final way and compute total of costs

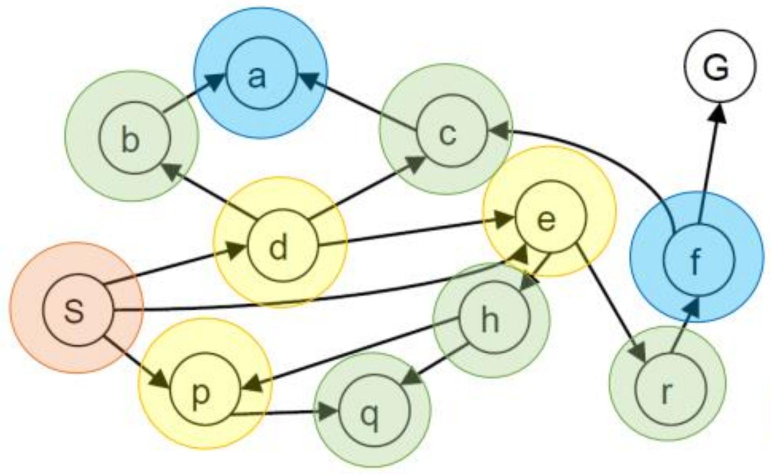
b. Example

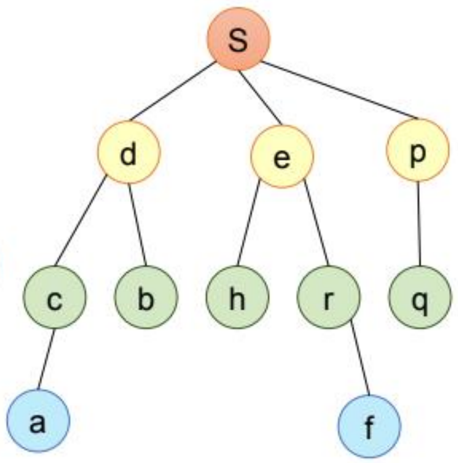


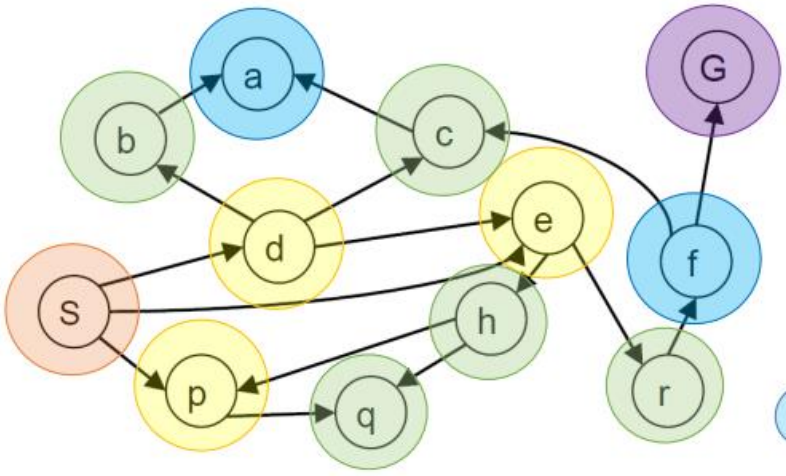


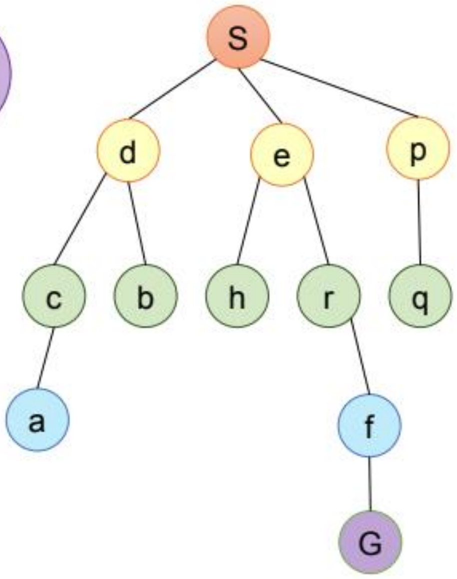

a
G
b
c
e
d
f
S
h
p
q
r


S
d
e
p
c
b
h
r
q

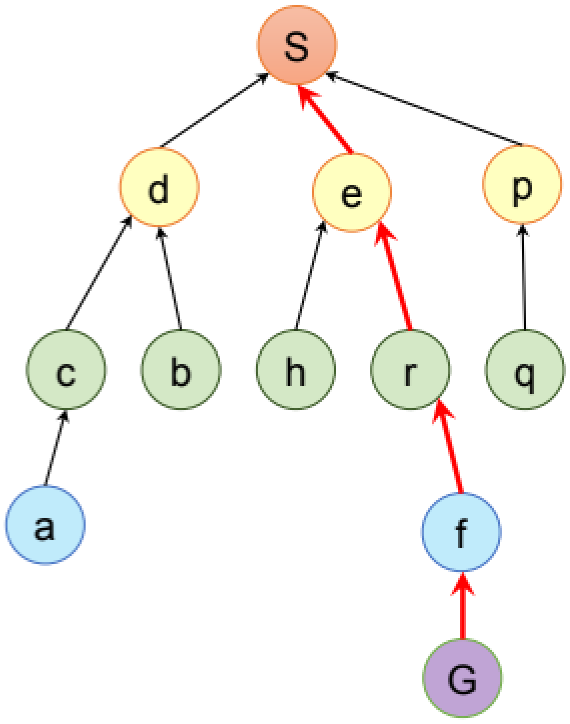

a
G
b
c
e
d
f
S
h
p
q
r


S
d
e
p
c
b
h
r
q
a
f





d = 4



[illegible]

c. Conclusion

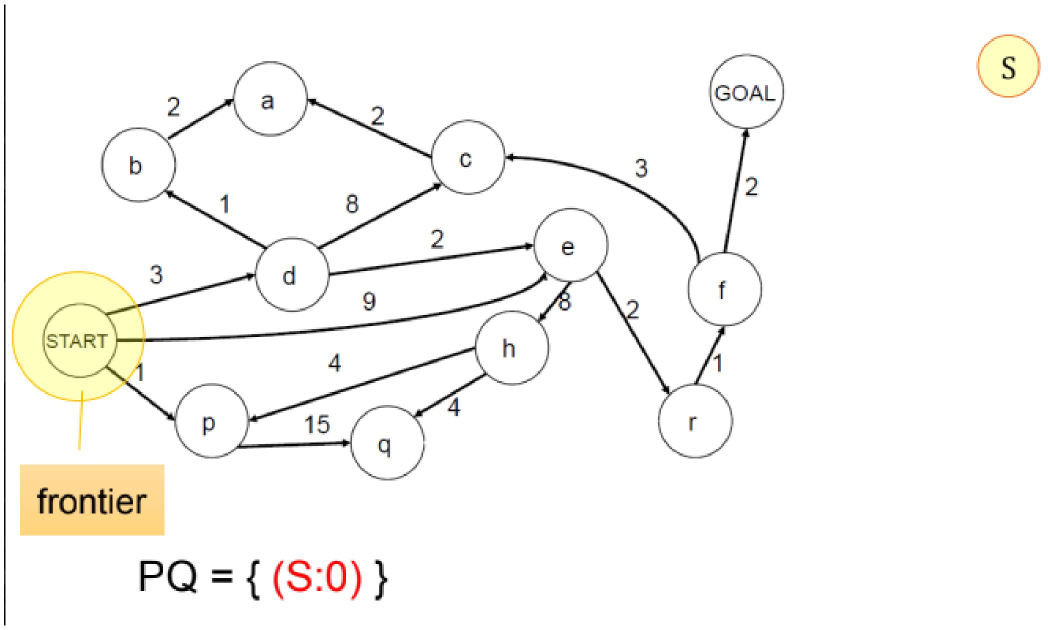
- It can use to find a way on map which doesn't have number about the distance between place to place but it still found the shortest way to go to the goal.
- Can't find the cheapest path

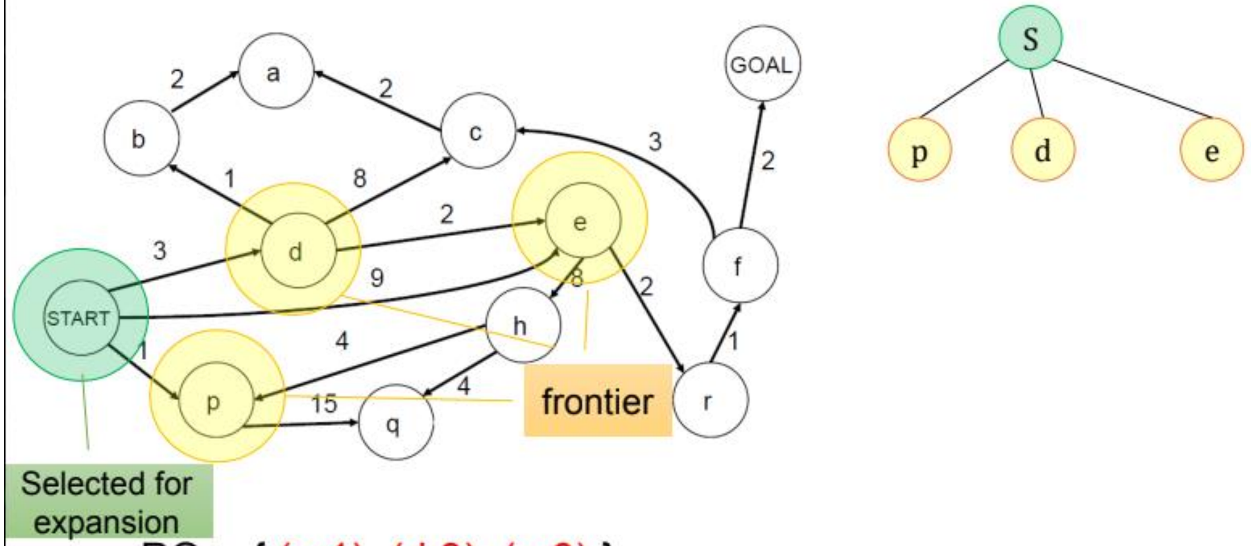
II. Uniform-cost search

a. Idea

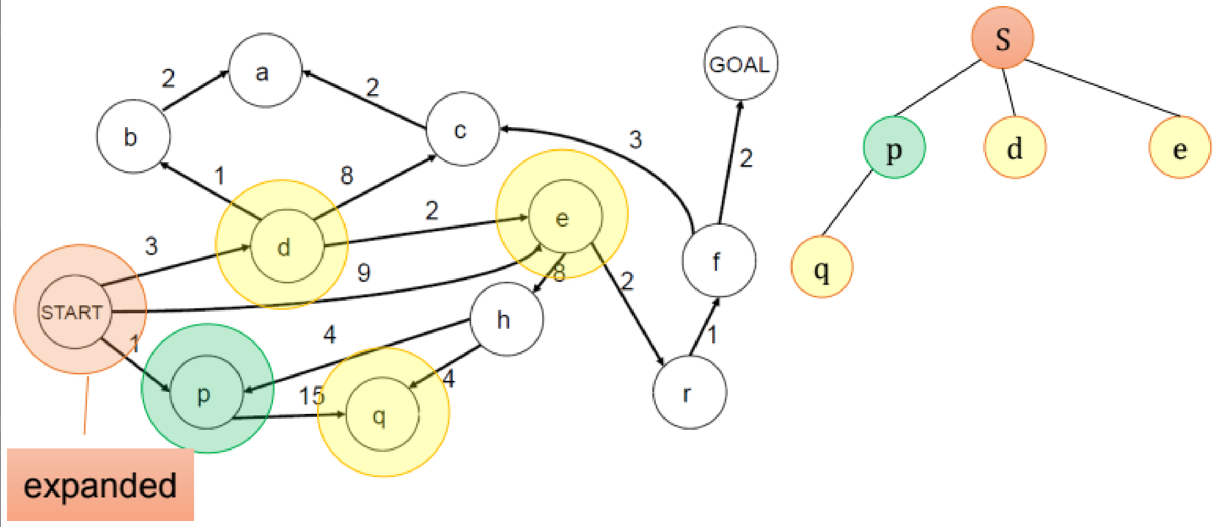
- At the same with breadth-first search but have value of the cost
- Update path cost
- Compare distance to find a next way need to go

b. Example

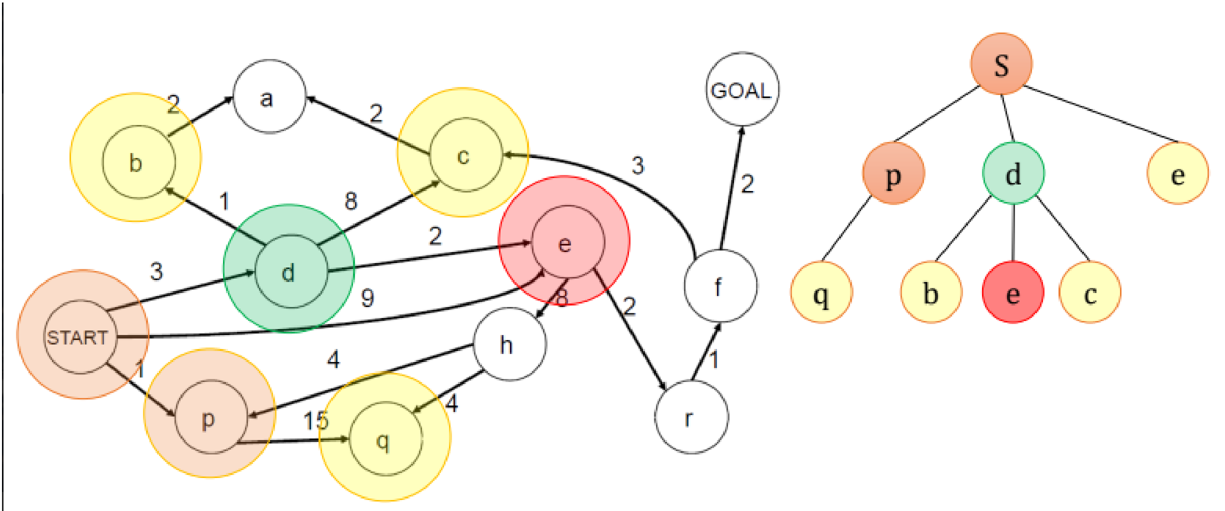




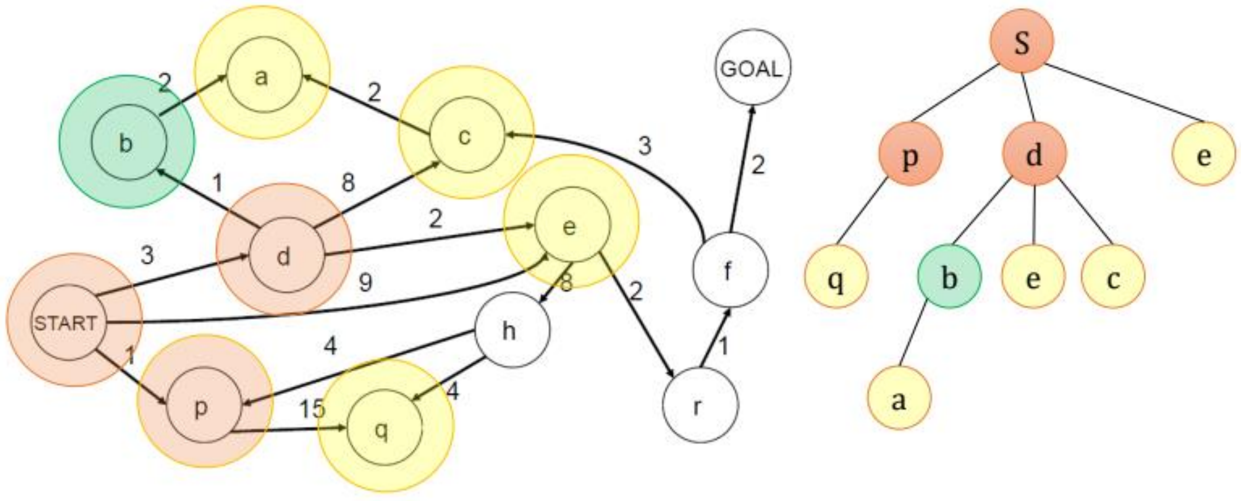
PQ = { (p:1), (d:3), (e:9) }



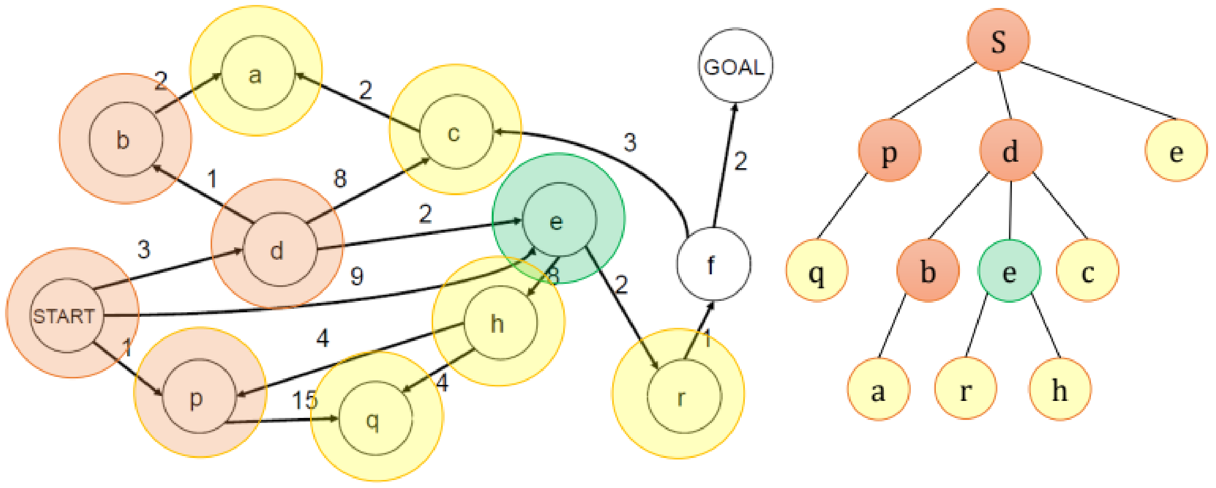
PQ = { (d:3), (e:9), (q:16) }



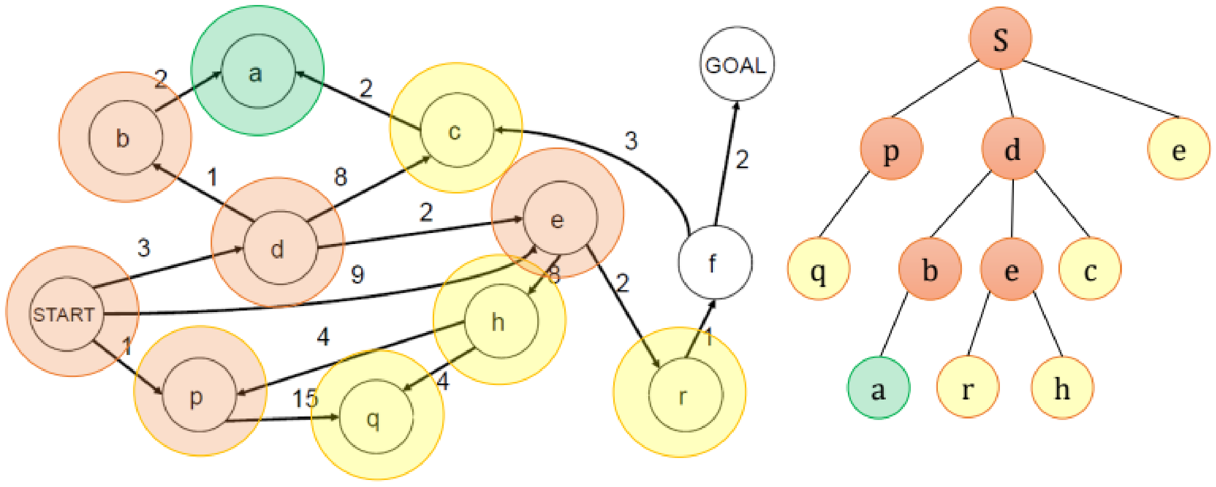
PQ = { (b:4), (e:5), (c:11), (q:16) }



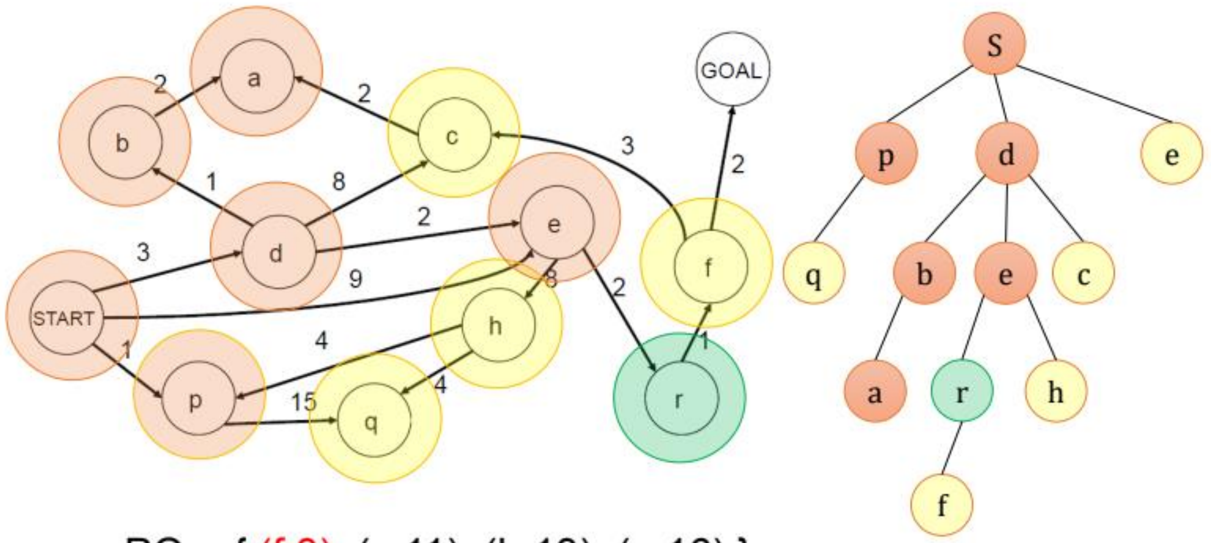
PQ = { (e:5), (a:6), (c:11), (q:16) }



PQ = { (a:6), (r:7), (c:11), (h:13), (q:16) }



PQ = { (r:7), (c:11), (h:13), (q:16) }



PQ = { (f:8), (c:11), (h:13), (q:16) }

Search path: S → d → e → r → f → G, cost = 10

c. Conclusion

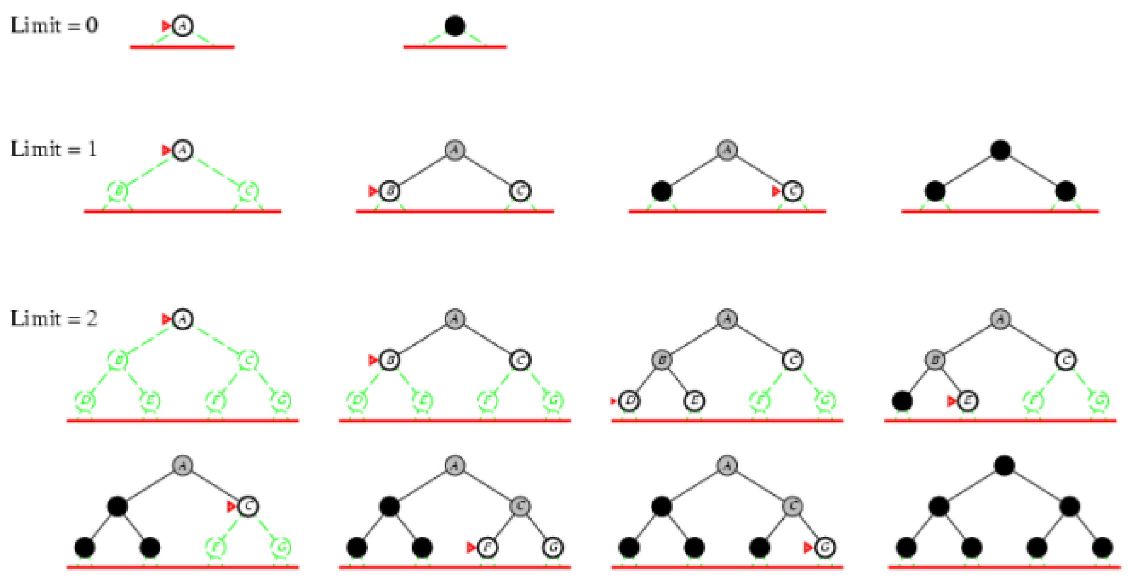
- It is optimal
- Find the cheapest path

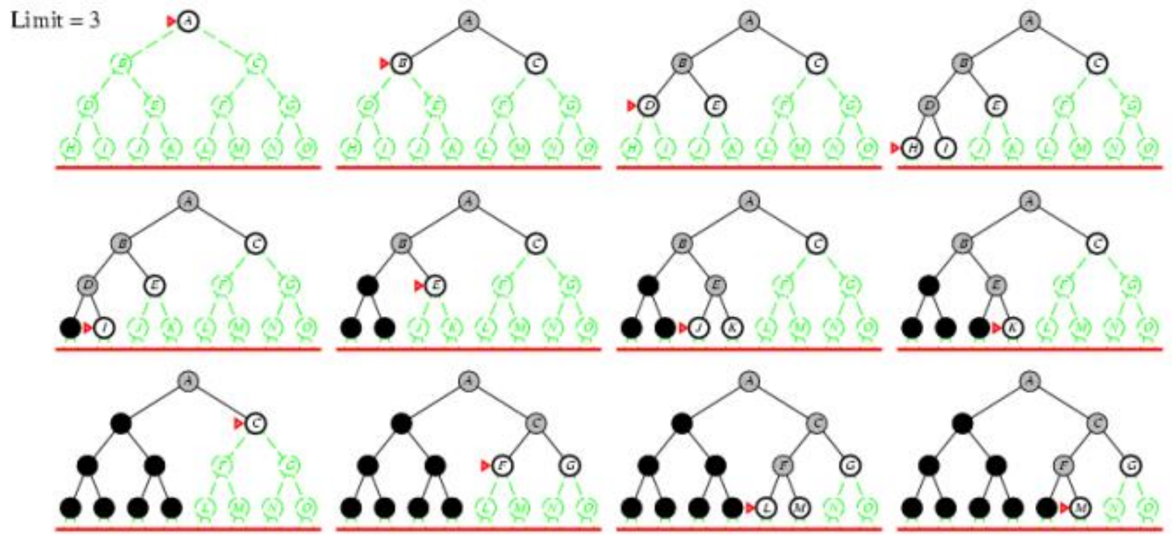
III. Iterative deepening search

a. Idea

- Movement same algorithm before
- Create the value contain the limit and depth
- Create function check that the depth is reach the limit or not

b. Example





c. Conclusion

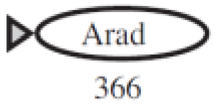
- Optimality
- Time complexity: $(d+1)b^0 + db^1 + (d-1)b^d = O(b^d)$
- Space complexity: $O(bd)$, similar to DFS
- Preferred when the search space is large and the depth of the solution is not known

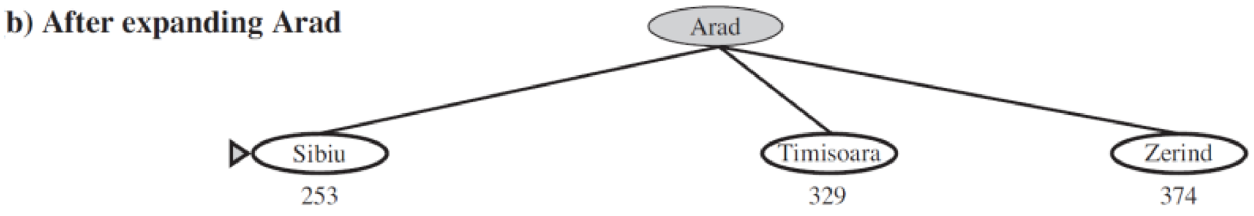
IV. Greedy best-first search

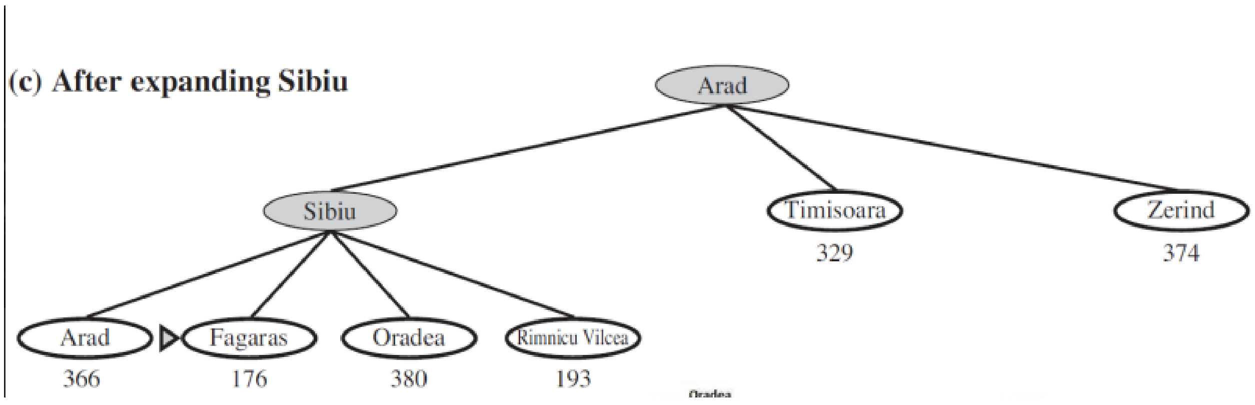
a. Idea

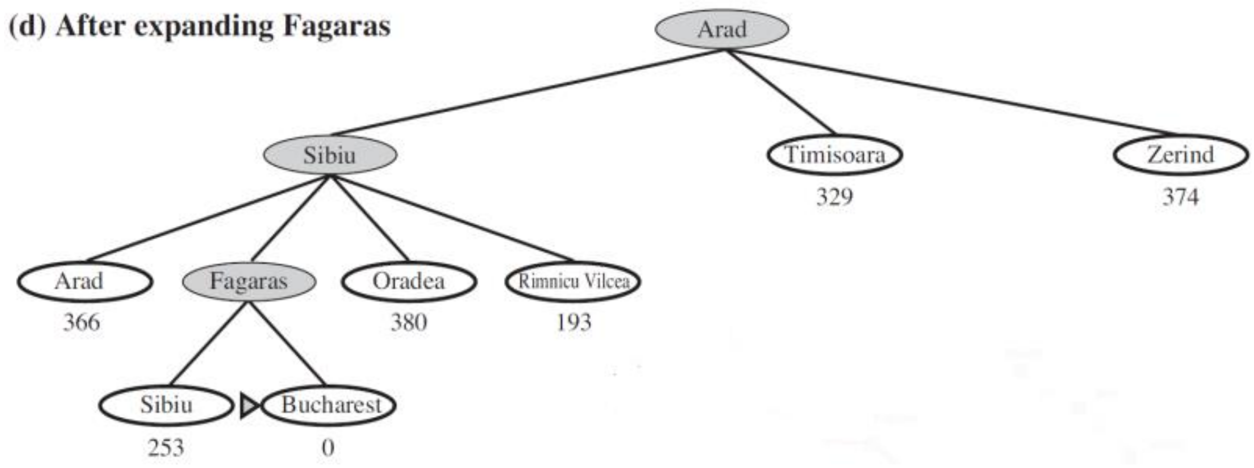
- Movement same another algorithm
- Bonus function to compute Manhattan
- Compare the distance and choose a the lowest

b. Example









c. Conclusion

- Time reduced substantially with a good heuristic
- Space complexity: $O(b^m)$ – keeps all nodes in memory
- No optimality

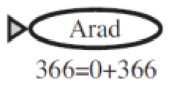
V. <u>Graph-search A*</u>

a. Idea

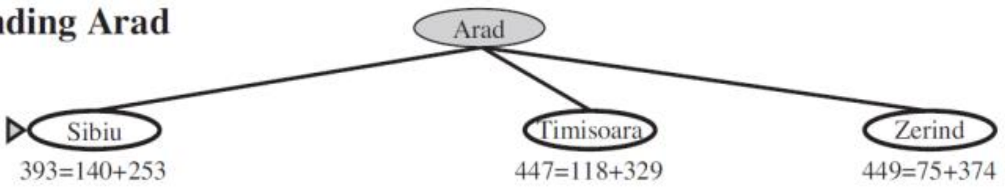
- Same Greedy best-first search
- More the value to contain cost
- Distance sum by Manhattan and the cost and compare it

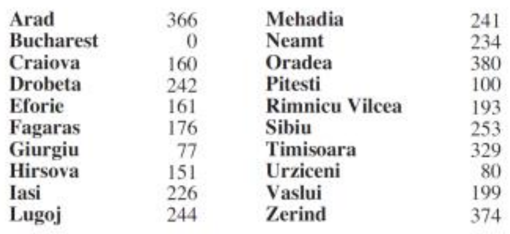
b. Example

(a) The initial state



(b) After expanding Arad





Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

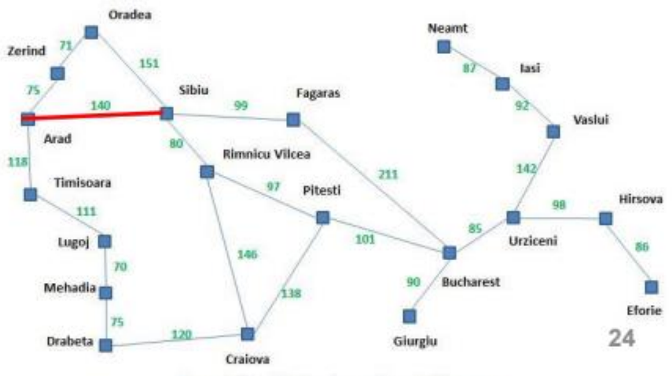
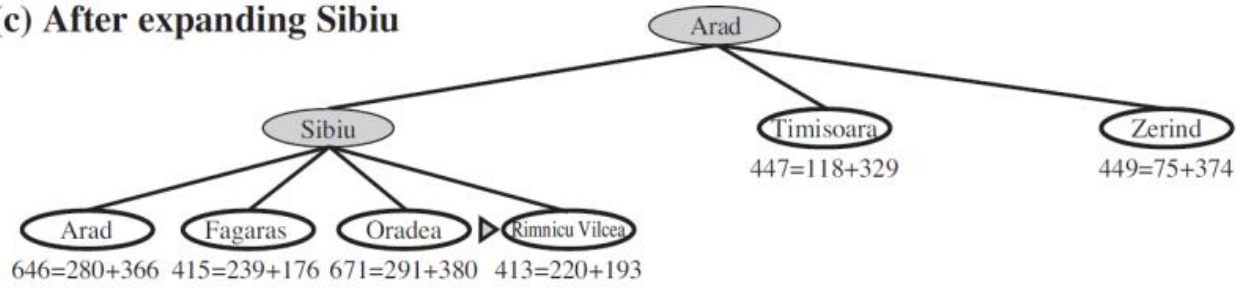
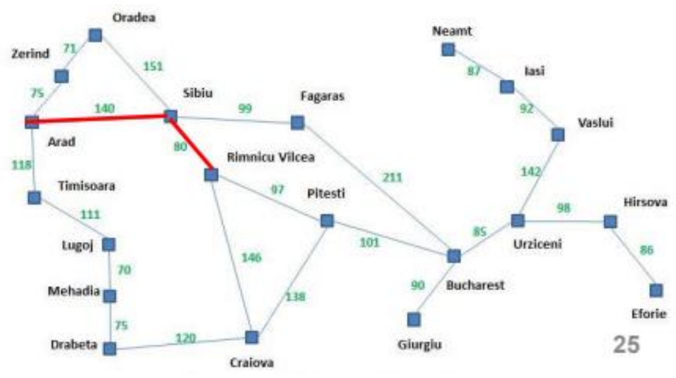


Figure A simplified road map of part of Romania.

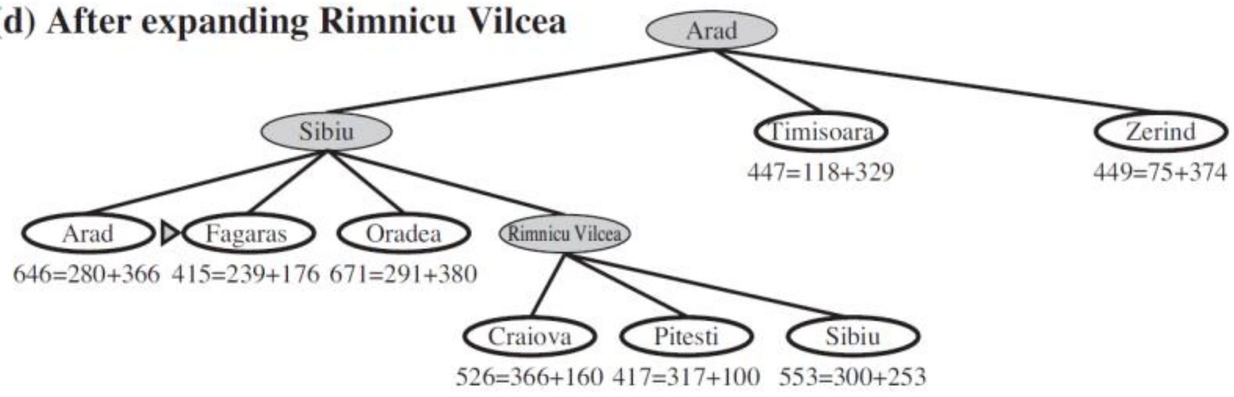
(c) After expanding Sibiu



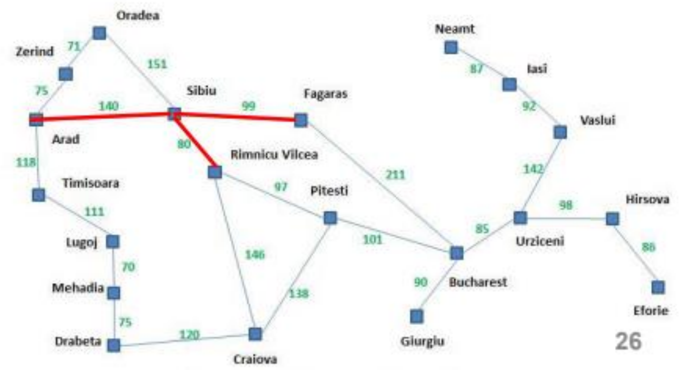
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



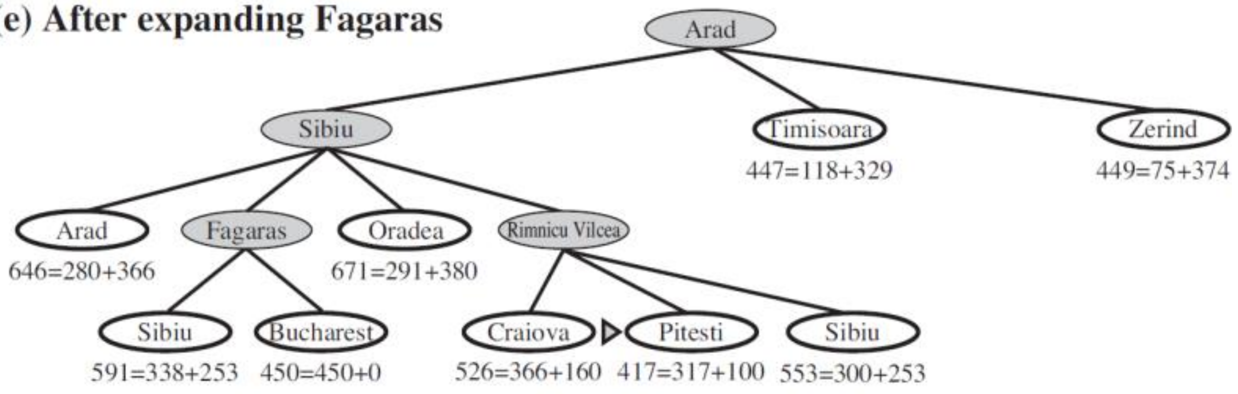
(d) After expanding Rimnicu Vilcea

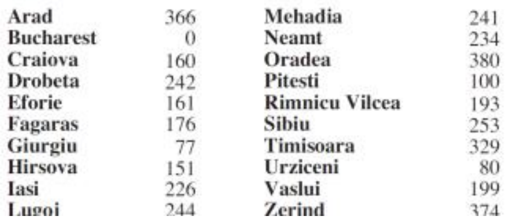


Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

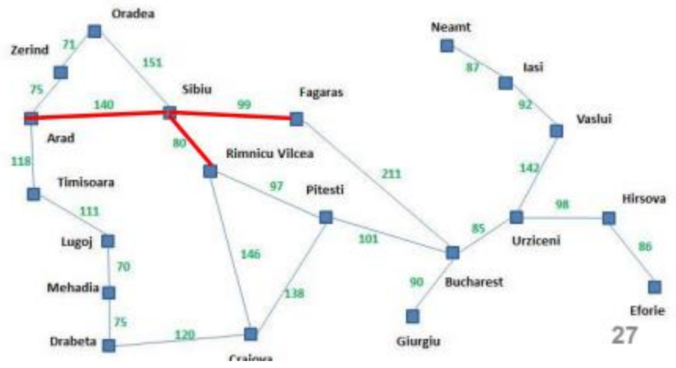


(e) After expanding Fagaras

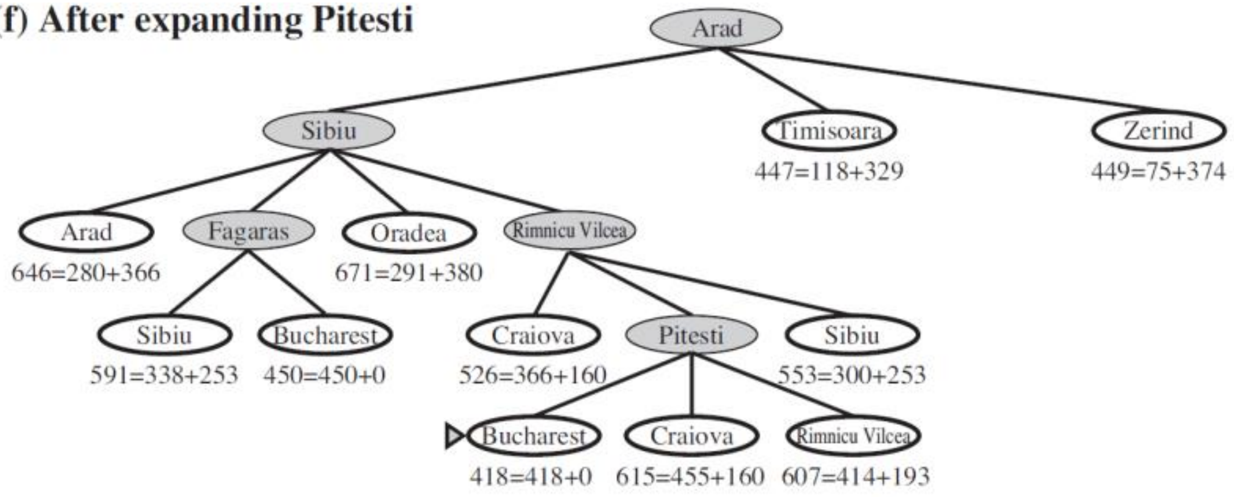


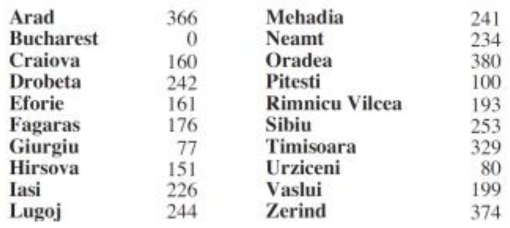


Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoi	244	**Zerind**	374

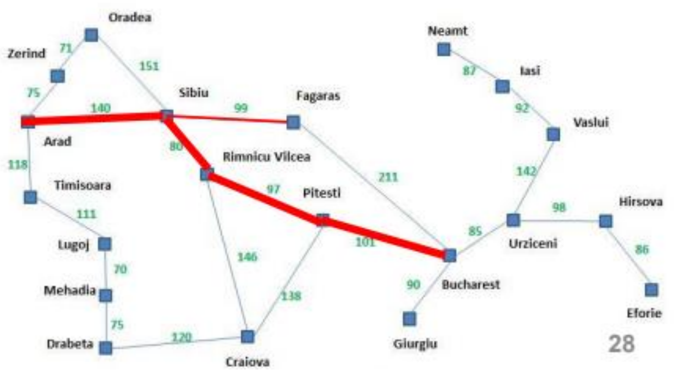


(f) After expanding Pitesti





Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



c. Conclusion

- Not always optimality
- Time complexity: exponential
- Space complexity: exponential

VI. References

- The document in the Computer Science Department at the University of Science, Vietnam National University, Ho Chi Minh City
- GeeksforGeeks
- StackOverflow