Vertex Cover problem and its solutions

Purpose

In this report, I am mainly focus on how to find minimum vertex cover on a randomly generated undirected graph. I used three algorithms to find the solution. the first one is **brute force**. The second one is **recursion**, and the last one is **approximation method**. For each method, I count the running time and compare with other methods in term of their time complexity.

Environment

In this report, I used java language and Intellij Idea IDE. There are 7 total java file, which are GraphNode.java, GraphEdge.java, Graph.java, GraphADT.java, GraphException.java, cover_vertex.java, findmini_vertex.java. The GraphNode.java, GraphEdge.java files define node class and edge class. Graph.java implement GraphADT.java. The **cover_vertex.java** file define the class who create graph in the constructor as its property and also define three algorithms as its methods(this class is most important class). the findmini_vertex.java has main function to generate random undirected graph with given nodes number. and then we try to test if the generated graph can have a vertex cover given k by using the brute force and recursion. Those two methods should return exact same result. Then we try to use polynomial-time "approximation algorithm" method to get the approximation solution.

Algorithm

At first algorithm, I used the brute force method, suppose we are given a graph $= \mathrm{G}(\mathrm{V}, \mathrm{E})$ and an integer k. The certificate we choose is the vertex cover $\mathrm{V}' \subseteq \mathrm{V}$ itself. The verification algorithm affirms that $|\mathrm{V}'| = \mathrm{k}$, and then it checks, for each edge $(\mathrm{u}, \mathrm{v}) \in \mathrm{E}$, that $\mathrm{u} \in \mathrm{V}'$ or $\mathrm{v} \in \mathrm{V}'$.

$Boolean\ bruteforce(G, k)\{$

1. $if(G\ contains\ no\ edges) return\ true$
2. $if\ (G\ contains \geq kn\ edges) return\ false$
3. $calculate\ all\ possible\ V' \in V\ and\ |\mathrm{V}'| = k$
4. $if\ exsit\ V',\ for\ all\ the\ edges(u, v) in\ G$
5. $\quad u \mid v\ \in\ V'$
6. $\quad return\ true$
7. $return\ false$
8. $\}$

(1.1)

The second algorithm is recursion, suppose we are given a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E}))$

And an integer k. when k = 0, the base function return true. otherwise we choose any edge (u, v) and delete two endpoints and its connected edge respectively from G. get G' and G'' and pass them with k-1 into two sub recursion function. The current function return true when at least one of its children functions return true.

Algorithm below is the pseudocode:

$boolean\ vertex\_cover(G, k)$

1. $if\ (G\ contains\ no\ edges)\ return\ true$
2. $if(G\ contains \geq k \cdot n\ edges) return\ false$
3. $Let\ (u, v) be\ any\ edge\ of\ G$
4. $a = vertex\_cover(G - \{u\}, k - 1)$ $remove\ node\ u\ and\ any\ edge\ connect\ with\ u$
5. $b = vertex\_cover(G - \{v\}, k - 1)$ $remove\ node\ u\ and\ any\ edge\ connect\ with\ v$
6. $return\ a\ or\ b$

(1.2)

The third algorithm is approximation algorithm, the algorithm, given the G(V,E), return the nodes which cover all edges. For each time the algorithm chooses a arbitrary edge(u, v). record u,v and delete every edge incident on either u or v from E. loop until the E is empty.

$APPROX - VERTEX - COVER(G)$

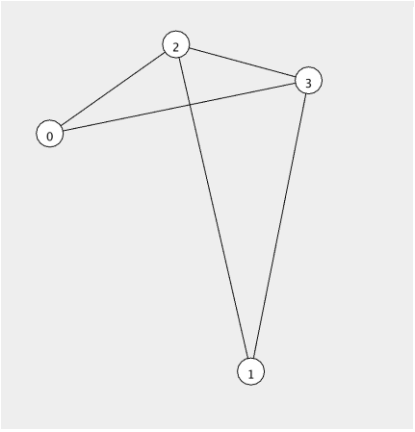
1. $C = \emptyset$
2. $E' = G.E$
3. $while\ E' \neq \emptyset$
4. $\quad let\ (u, v) be\ an\ arbitrary\ edge\ of\ E'$
5. $\quad C = C \cup \{u, v\}$
6. $\quad remove\ from\ E' every\ edge\ incident\ on\ either\ u\ or\ v$
7. $return\ C$

(1.3)

Complexity analysis

let us randomly generated a undirected graph with n = 4, 6, 8, 10 then we try to give k = 2,3,4,5 then return if the graph has less or equal than k vertex cover. also, we try to find the approximation solution using the approximation algorithm. Eventually, we compare the running time and calculate the time complexity for all the scenario and different algorithms.

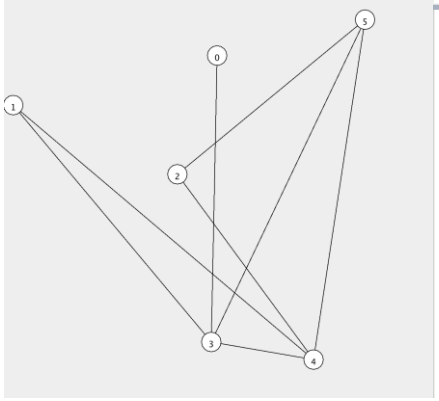
At first we generate for n = 4, k = 2 graph.




```
/Library/Java/JavaVirtualMachines/jdk-10.jdk/Contents/Home/
the current k is 2
true
the runing time for bruteforce is  319008
the bruteforce solution  is 2 3
true
the runing time for recursion is  274953
the recursion solution is 2 3
the runing time for appximation method is  38703
the approximation solution is 1 2 0 3
```

Here we see that brute force and recursion algorithm both return the node 2,3. Compared with the approximation algorithm, who return all nodes. Since the approximation algorithm is polynomial-time 2-approximation algorithm. Which means the optimal solutions is at least 2. And given we found solutions when k = 2. So, k = 2 is optimal vertex cover.

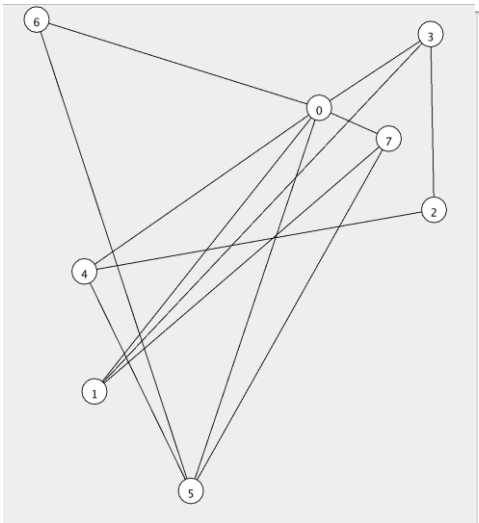
For $n = 6, k = 3$




```
/Library/Java/JavaVirtualMachines/jdk-10.jdk/Contents/Home/bi
true
the runing time for bruteforce is  581317
the bruteforce solution  is 2 3 4
true
the runing time for recursion is  441245
the recursion solution is 3 4 5
the runing time for appximation method is  59463
the approximation solution is 5 1 2 3 0 4
```

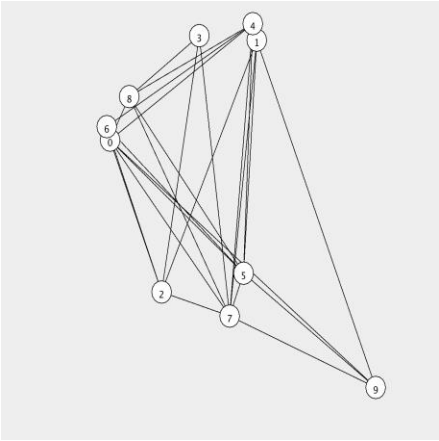
Same as previous situation, the approximation solution return 5,1,2,3,0,4 those six nodes. Which means that the optimal k must be at least 3. And we find solution when k = 3. So, k = 3 is optimal vertex cover.

We can see that for n =8, k =4




```
/Library/Java/JavaVirtualMachines/jdk-10.jdk/Conten
the current k is 4
true
the runing time for bruteforce is  989288
the bruteforce solution  is 0 1 2 5
true
the runing time for recursion is  549657
the recursion solution is 0 1 2 5
the runing time for appximation method is  63657
the approximation solution is 0 3 5 2 1 4
```

We can see that for n= 10, k =5.




```
/Library/Java/JavaVirtualMachines/jdk-10.jdk/Contents/Home/bin/j
the current k is 5
false
the runing time for bruteforce is  4507750
false
the runing time for recursion is  1398855
the runing time for appximation method is  99428
the approximation solution is 3 2 4 6 0 7 5 1
the current k is 6
false
the runing time for bruteforce is  1188998
false
the runing time for recursion is  95408699
the runing time for appximation method is  39228
the approximation solution is 3 2 4 6 0 7 5 1
the current k is 7
true
the runing time for bruteforce is  457644
the bruteforce solution  is 0 1 2 3 4 5 7
true
the runing time for recursion is  1794669
the recursion solution is 2 4 5 6 7 8 9
the runing time for appximation method is  31878
the approximation solution is 3 2 4 6 0 7 5 1
```

When n = 10, we first input k = 5, and both algorithms return false, then we try to increase k iteratively until both algorithms return true. it is the time when k = 7. So, we say 7 is the minimal vertex cover.

From those four graphs, we can roughly see that the running time for brute force algorithm is longest, the recursion is slightly better(but not beat brute force in every cases) and the running time for approximation algorithm is shortest. and also as n increase from 5 to 10, the brute force and recursion's running time increase dramatically over 10 times, however the approximation algorithm increase slowly(almost linearly with n). Also it is worthy to notice, compared with other two algorithm, it also returns the most nodes. it is because the positive correlation between the time spend and the accuracy of solutions. We will do the time complexity analysis to explain this later.

In term of the time complexity of brute force algorithm(1.1). We first got the all combination of k nodes out of n nodes and add them into hash set. The time complexity for this procedure is $c_1 \cdot \binom{n}{k}$, so from line 1-3 time complexity is $c_1 \cdot \binom{n}{k}$. Then we iterate all edges for each V'. the size of V' is $\binom{n}{k}$, assume the size of edge is E. so from line 4-7, the time complexity is $c_2 \cdot E \cdot \binom{n}{k}$ And also the edge number must less than kn, so we can use its upper bound $c_2 \cdot k \cdot n \cdot \binom{n}{k}$ to replace it .combined line 1-7 together, the total time complexity is $c_1 \cdot \binom{n}{k} + c_2 \cdot k \cdot n \cdot \binom{n}{k}$. Since the k is not a constant. The algorithm is not polynomial solvable.

In term of recursion method(1.2), since there are less and equal to 2k+1 nodes in the recursion tree; each invocation takes $O(kn)$ time. So, the algorithm has time complexity of $O(2^k \cdot kn)$. Since the k is not a constant. The algorithm is not polynomial solvable.

$$T(n,k) \leq \begin{cases} c & \text{if } k = 0 \\ cn & \text{if } k = 1 \\ 2T(n,k-1) + ckn & \text{if } k > 1 \end{cases} \Rightarrow T(n,k) \leq 2^k ckn$$

For the approximation method(1.3). eventually we are going to remove all the edges from E' and add some nodes to C. we can easily deduct that the time complexity is O(V+E),since V and E can both write in the polynomial of O(n). So, the time complexity is still O(n).so we can see the running time is almost increase linearly with n.

Discussions

We know the vertex cover problem is NP problem. which means that given a certificate, we can verify it in polynomial time. however, since it is NPC problem, it is hard an algorithm that is polynomial solvable.as we have already show in previous section, the time will become a huge issue when n grow for backtracking and recursion, so we can use the approximation algorithm to find the "approximation solution" in much less time(almost linear). It works for some cases that we don't need accurate solutions but we are strictly time constrained.

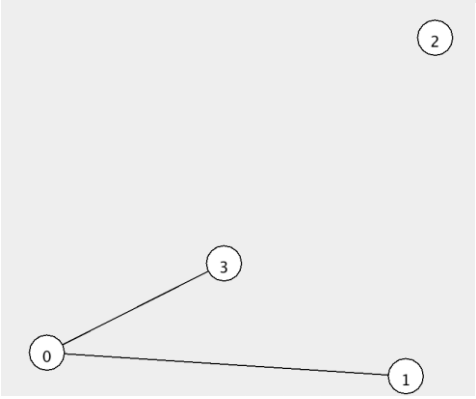
How to run the given codes?

first unzip the assignment 5 file, and you will get 7 java files. You should put them into the same folder and use IDE to open them.

The main function is findmini_vertex.java, in this file, you can input any array of n and k as long as $k \leq n$ and n.length = k.length.

```
public class findmini_vertex {
    public static void main (String[] args) throws GraphException {
        int[] n = new int[]{4};
        int[] k = new int[]{2};
```

Then you will get the output and display the graph. the sample output will be like this.




```
/Library/Java/JavaVirtualMachines/jdk-10.jdk/Contents/Home/b
the current k is 2
true
the runing time for bruteforce is  199477
the bruteforce solution  is 0 1
true
the runing time for recursion is  281514
the recursion solution is 1 3
the runing time for appximation method is  28277
the approximation solution is 1 0
```

Reference

Kevin Wayne, 2005, https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pearson/10ExtendingTractability-2x2.pdf

"Introduction to Algorithms". MIT Press. *Retrieved 2017-07-02*.

Knuth, Donald (1974). "Postscript about NP-hard problems". ACM SIGACT News. ***6*** *(2): 15–16. doi:10.1145/1008304.1008305.*