CS4234: Optimisation Algorithms Problem Set 3

PS3 - Mini Researcher

V1.2: Steven Halim *September 19-October 08, 2018*

Preliminaries

I will start PS3 by reviewing two parts in Lecture 0 rules for the purpose of this PS3 - Mini Researcher.

1. You can search, but 'do not ask for solutions on the web'. That is, you are allowed to browse the Internet for existing content as of September-October 2018 but you just not allowed to actively post questions to general public hoping someone out there will accidentally give answers (or hints) for you.
2. If you discuss with friends, 'just take general notes during such discussions'. Ultimately, you need to write the answers yourself, in your own words. Referring to other people's writeup will very likely result in you writing very similar writeup.

To-Do

Do the following tasks below and submit a written (or typed) report of **no more than SEVEN (7) pages**... That is, Yohanes (main grader) and myself (I will also read the more interesting answers myself afterwards after Yohanes summarizes his initial grading) will only read at most 26*7 = 182 pages for this PS3.

Please use reasonable font. Illegible handwriting or minuscule font sizes will get penalty.

Submit the **hardcopy** to Gan Wei Liang during T05 on Monday, 08 October 2018. The hardcopy will be manually :O graded by Yohanes and then myself (not via Mooshak this time) in about 2 weeks time.

Tasks

There are 4 tasks with total of 100 marks that will be scaled back to 5% for this PS3.

Task 1 (20 marks): In Lecture 4, I mentioned a fictional movie about P = NP and its implication (Travelling Salesman, 2012, `http://www.travellingsalesmanmovie.com/`). Some other people have imagined and written books or articles on what our world will be like if someone eventually proves P = NP (or otherwise), e.g. The Golden Ticket (`http://goldenticket.fortnow.com/`). In this task, I want you to imagine both sides and then write down your imagination of what may happen if someone proves P = NP (12 marks; looking for creative answers, hopefully more creative than in the past two AYs) and what may happen if someone proves P $\neq$ NP (the other 8 marks, as nobody has conclusively proven either way).

The world if P = NP	**The world if P $\neq$ NP**
My imagination is (remember 7 pages limit)...	My imagination is (remember 7 pages limit)...

Task 2 (40 marks): In Lecture 1+2, I introduced:

- The Vertex-Cover problem (the NP-complete decision version) and show quick reduction from Clique problem,
- The Min-Vertex-Cover problem (the NP-hard optimization problem), and
- The Min-Weight-Vertex-Cover (the weighted variant).

Then we went on and discussed:

- Several special cases of the MIN-VERTEX-COVER problem that have polynomial solutions (lose generality): On Binary Tree, On General Forest of Trees (PS1A), on Graph with maximum degree-2 (T01), etc...,
- Several parameterized (brute force) solutions (lose speed): The $O(2^k \times m)$ Divide and Conquer like algorithm (PS1B default), the $O(2^n \times m)$ brute force algorithm, and certain other non-discussed publicly techniques (also possible for PS1B)...,
- Several approximation algorithms (lose optimality): The 2-approximation algorithms: Randomized (but only 2-opt *in expectation*) and Deterministic using Matching (PS1C), and the 2-approximation algorithm using relaxed ILP for the weighted variant and also the Bar-Yehuda and Even's (or pricing) algorithm (PS1D that can also solve PS1C).

We have not discussed max-flow/matching based algorithms or local search algorithms for these problems and will do so only in the last few lectures of CS4234. For now, your job is to research (Internet, books, etc) on **another** NP-hard Optimization Problems and report your findings.

The choice of that other NP-hard Optimization Problems for this semester is restricted to the following: PARTITION (e.g. read `https://en.wikipedia.org/wiki/Partition_problem`), PARTITION-INTO-TRIANGLES (e.g. read `http://www.cs.ucsb.edu/~teo/cs230.f14/npcpit.pdf`, now I realize why my role as ICPC coach is super hard when I have to form strong teams), or SATISFIABILITY (e.g. read `https://en.wikipedia.org/wiki/Boolean_satisfiability_problem`, various variants exists). Please superficially explore all three problem names first, and then after roughly understand the options, pick **just one** that you like the most and zoom in into that problem as your report for this Task 2. The best reports on this task may be scanned, compiled, and distributed for self-learning for all of us, including myself (and future CS4234).

My selected NP-hard Optimization Problem to be self studied is...
Write the name of your selected problem here...
Can you illustrate that problem in layman terms?
Like Opening (Coffee Shop) branches for MIN-VERTEX-COVER...
Can you prove the NP-completeness of the decision version of that problem?
Write your short proof here (see Lecture 1/CS3230), remember that you only have 7 pages...
Can you find any special cases about this problem that has polynomial solution?
Mention at least one and the expected polynomial solution, the more interesting, the better...
Can you find some heuristics/pruning strategies to brute force this problem?
Mention at least one, the more interesting, the better, mini experiment is better...
Can you find any approximation algorithm for this problem?
Mention the algorithm (and its name, if any) and prove the approximation ratio...

Task 3 (20 marks): In Lecture 5+6 combo, we review *various* implementation of Ford-Fulkerson's method: Basic (quick review), Fattest Path (skipped), Capacity Scaling (skipped), Edmonds Karp's (detail), and Dinic's (quick review) that differ mostly on the way they find augmenting paths. In Lecture 7, we will learn another MAX-FLOW algorithm: Push-Relabel. For now, let's just concentrate on one implementation of Ford-Fulkerson's method: The Edmonds Karp's algorithm that has $O(m^2 \times n)$ or $O(VE^2)$ time complexity that we analyzed nearing the end of Lecture 5+6 combo.

First download a rather 'slow' $O(VE \times V^2) = O(V^3 \times E)$ implementation of Edmonds Karp's algorithm inside **ch4.zip** from `https://cpbook.net/#downloads` (Competitive Programming 3). There are both ch4_08_edmonds_karp.cpp and ch4_08_edmonds_karp.java in that zip folder. Now please add an additional counter inside the main loop, i.e. from this C++ code:

```
  mf = 0;
  while (1) {              // O(VE^2) (actually O(V^3E) Edmonds Karp's algorithm
    f = 0;

    // Run BFS, details hidden

    augment(t, INF);
    if (f == 0) break;
    mf += f;
  }

  printf("%d\n", mf);
```

Into:

```
  mf = 0;
  int NumAugmentingPaths = 0; // add this counter
  while (1) {              // O(VE^2) (actually O(V^3E) Edmonds Karp's algorithm
    f = 0;

    // Run BFS, details hidden

    augment(t, INF);
    if (f == 0) break;
    mf += f;
    NumAugmentingPaths++; // we run one more iteration of EK's algorithm
  }

  printf("EK finds %d augmenting paths throughout its execution\n", NumAugmentingPaths);
```

Now your main job for this task is to manually create one test case — a flow graph — of *at least* $V = n = 7$ vertices (and $E = m \geq 6$ edges) that can force that Edmonds Karp's implementation to find as close to $VE = nm = 42$ iterations as possible. For example, giving this test case to my Edmonds Karp's implementation:

```
7 0 6
1   1 7
1   2 17
1   3 27
1   4 37
1   5 47
1   6 57
0
```

will clearly just give 1 augmenting path with bottleneck edge weight 7, so the max flow value is clearly 7. So construct a worst test case that forces Edmonds Karp's to work the hardest, i.e. at $O(VE)$ iterations/augmenting paths. Explain why your worst test cases force Edmonds Karp's to behave that way. Near full marks is given to student who can come up with the 'current best' (that is, the 'current worst') test case and full marks if the student can also explain why that test case is the hardest possible for Edmonds Karp's.

Task 4 (20 marks): Starting from last AY 2017/2018, a few CS4234 content are available in VisuAlgo with help of Rais (CS4234 TA and Steven's FYP student last AY). However, now the progress is currently on hold as there is no current FYP student working on VisuAlgo. For the first 4 lectures about various

NP-hard Optimization problems, we have `https://visualgo.net/en/mvc`, `https://visualgo.net/en/steinertree`, and `https://visualgo.net/en/tsp`. We are still lacking visualization for Min-Set-Cover and a few others that we discussed in tutorials though, like Max-Clique, Min-Feedback-Edge-Set etc...

There are two visualization modules in VisuAlgo for the next 4 lectures: `https://visualgo.net/en/maxflow` and `https://visualgo.net/en/matching`. However, the Max Flow visualization currently lacks algorithm like Push-Relabel and can probably be made much clearer. Also, the Graph Matching visualization is currently restricted to the unweighted Max Cardinality (Bipartite) Matching problem (no weighted variant yet... I am particularly interested in implementing animation of Hungarian (weighted matching) algorithm).

Finally, there is no proper visualization for the last 4 lectures of CS4324 as my old local search visualization work for his PhD `http://www.comp.nus.edu.sg/~stevenha/viz/` is currently inactive :(.

In this task, your job is to scour the Internet to check what has been done by *other* people out there about the topics discussed in the first 6 lectures of CS4324 so far (i.e. has anyone done visualization on Min-(Weight-)Vertex-Cover, Max-Clique, Min-Set-Cover, Min-(Weight-)Feedback-Edge-Set, etc?) and then give suggestions to further improve VisuAlgo based on what others have done or from your own personal experience using these 5 VisuAlgo pages (mvc, steinertree, tsp, maxflow, matching). Note that only some of the better ideas will be eventually implemented in the near future (and only if I has time or I get new FYP student(s) working on VisuAlgo or I suddenly get significant teaching grant that allows me to hire workers/coders to actually implement these grand ideas).

My findings about what are currently available out there are...
Write your findings here...
My suggestions to improve VisuAlgo (especially mvc, steinertree, tsp, maxflow, matching) are...
Write your suggestions here... Format: If you find a glaring bug (with hopefully easy fix), tell us how to replicate it so that we can swiftly kill it If you want to suggest improvements, show us how to visualize the content in a better way (feature updates will take lots of time before it gets implemented)