This assignment is due on April 24 and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go [illegible] nd read the section: "Advice on how to do the assignment".

This assignment is [illegible] 7 and COMP3927 students. COMP3027 students should do Problems [illegible] 1P3927 students should do Problems 1 and 3.

Problem 1. (20 points)

To begin, let us consider the Ford-Fulkerson algorithm. Consider below the flow network G, c in Figure 1 and the current $s-t$ flow f in Figure 2. **Your task** is to run one iteration of the Ford-Fulkerson algorithm on the flow network.

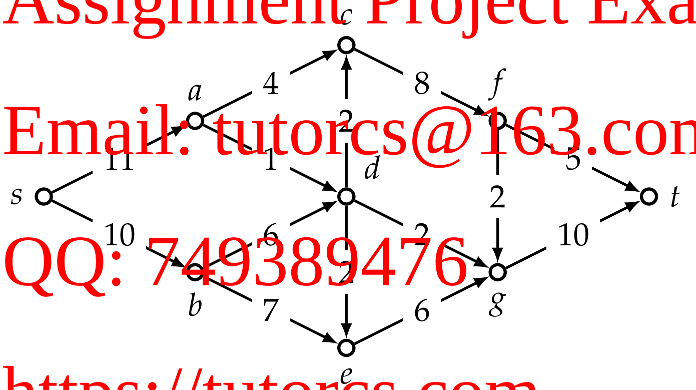

Figure 1: Flow network G with $c : \mathbb{E} \to \mathbb{Z}^+$ (edge capacities)

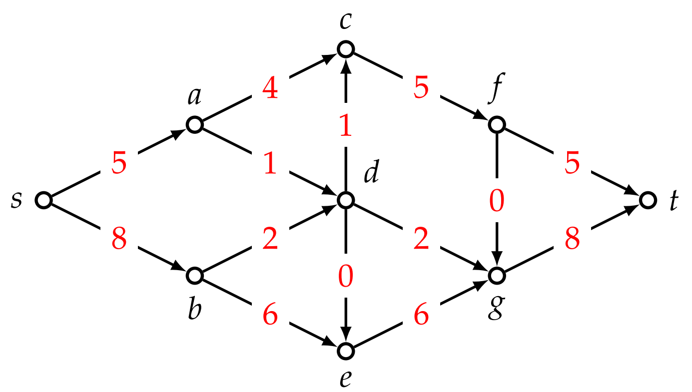

Figure 2: $f : \mathbb{E} \to \mathbb{Z}^+$ (current flow through G)

a) Draw the residual graph associated with the flow f. Specify the residual capacity of every edge. You are allowed to draw this by hand and insert a photo (Alternatively, there is a LaTeX template on the last two pages of this assignment [illegible])

b) Specify an s[illegible]pdate the flow.

c) After the u[illegible]low through the network a max flow? Explain why.

d) Specify a M[illegible]with graph G.

Problem 2. (80 points) **(COMP3027 only)**

You and your friends have decided to go on a great go-karting adventure. You've heard that there is to be a symposium for algorithms and computing theory (SACT for short) in the bush (with music!), and since your good friend André owns k go-karts, you decide that the most fun way to get there is via an epic four wheel journey. There is only one issue - the go-karts can only go so far before they need to be refueled, meaning you need to design a trip that allows for picking up fuel every 10km. Even worse, you have realised that all these stops only have enough fuel for one go-kart, meaning that each individual go-kart will need to refuel at different stops. Formally, the *SACT Go-Kart Problem* is the following; given a positive integer k, n (2D) Cartesian points describing fuel stations and both a starting point S and end point T in the Cartesian plane, determine if k friends have valid paths from S to T. Note that a valid path involves stopping at a fuel station every 10km or less.

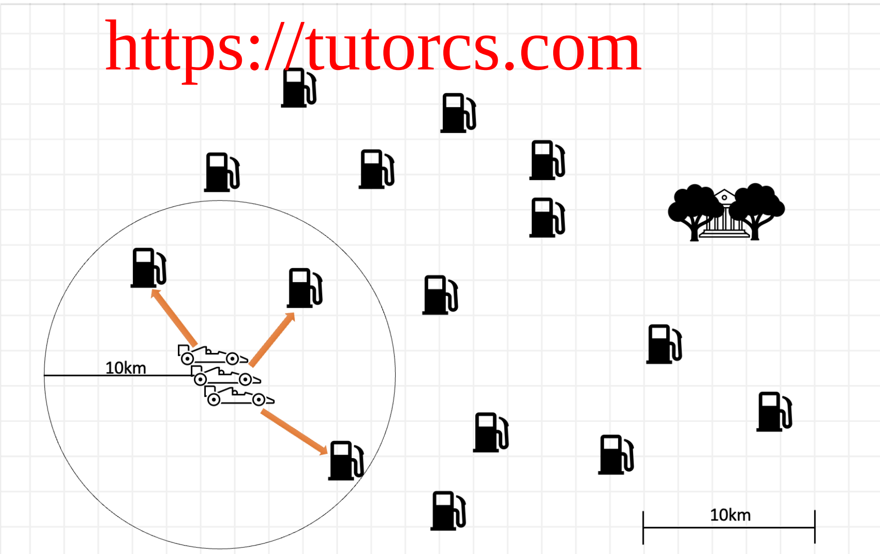

Figure 3: Map including go-karts start location, fuel stations, and the bush symposium. A go-kart cannot travel 10km without visiting a fuel station.

As a keen algorithm's student (after all, you're trying to get to an algorithms conference!), you realise that the SACT Go-Kart Problem can potentially be solved through the use of flow network algorithms!

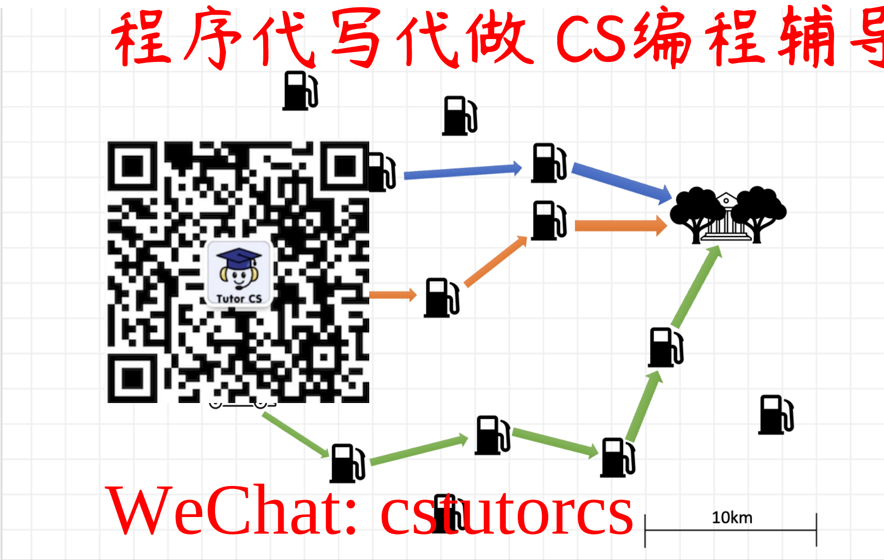

Figure 4: Map showing paths the go-karts could take to get to the symposium. Note that a fuel station can only be visited by one car, making three the highest number of paths from the starting location to the bush symposium.

Your task is to design a reduction from the SACT Go-Kart Problem above to an instance of Max Flow. For full marks, your reduction should be as fast as possible.

a) Describe how you translate an instance of the SACT Go-Kart Problem into an instance of Max Flow. In particular, you need to describe how you construct the flow network H based on the input of fuel stations, start and end points.

b) State which algorithm you use to compute the max flow in H.

c) Describe how you translate the output of the algorithm in (b) into a solution for the SACT Go-Kart Problem. In particular, describe how you determine if all k people are able to make the bush conference, and what their paths would be.

d) Prove the correctness of your reduction. In particular, you need to prove that there exists an integer F for which the following two statements hold:

- If the max flow is at least F, then your translation procedure in (c) produces feasible paths for all k people.
- If there are feasible paths for all k people, then the max flow is at least F.

e) Prove the time complexity of your entire reduction, taking into account the steps in parts (a), (b) and (c). Make sure that your time complexity is stated in terms of n and k.

Problem 3. (80 points) (COMP39[illegible] only)
People have caught wind of your algorithm for getting to the SACT in the bush, and others want to join your go-karting expedition (see Problem 2 for context). You decide to investig[illegible]-karts you can get access to for the event, and find that if peopl[illegible] to different starting locations, many more people will be able to [illegible]onference. Furthermore, you have been granted the ability to add [illegible]ing stations, meaning multiple karts can now visit the same sta[illegible]ou're now in charge of optimally adding this extra fuel to maxi[illegible]of algorithm-enthusiasts who can make it to the SACT bush confe[illegible]

Formally, we define the *p-SACT Go-Kart Problem* as the following; given a positive integer p representing the amount of extra fuel you can add to stations, n (2-dimensional) Cartesian points describing fuel stations, ℓ starting points S_i (with a positive integer k go-karts each) and an end point T all also in the Cartesian plane, determine the maximum number of valid paths starting from some $S_i (i \in \ell)$ ending at T. Note that 1 unit of fuel allows one more kart to pass through a given station. You wish to a) **maximise** the number of people who can get to the bush conference from starting points (*without adding fuel to stations*), and b) **minimise** the amount of fuel used to add an extra path from a start location to the end, before c) returning the paths they could take and where the fuel has been added.

Your task is to design a reduction from the p-SACT Go-Kart Problem above to an instance of Max Flow. For full marks, your reduction should be as fast as possible.

a) Describe how you translate an instance of the p-SACT Go-Kart Problem into an instance of Max Flow. In particular, you need to describe how you construct the flow network H based on the input of fuel stations, start and end points.

b) State which algorithm you use to compute the max flow in H.

c) Describe how you translate the output of the algorithm in b) into a solution for the p-SACT Go-Kart Problem. In particular, describe how you determine how many people are able to make the bush conference, and what their paths would be (do not optimise for p here).

d) Prove the correctness of your reduction. In particular, you need to prove that there exists an integer F for which the following two statements hold:

- If the max flow is at least F, then your your translation procedure in (c) produces feasible paths for K people.
- If there are feasible paths for K people, then the max flow is at least F.

You should also argue that you have used a minimal amount of fuel for this extra path.

e) Prove the time complexity of your entire reduction, taking into account the steps in parts (a), (b) and (c). Make sure that your time complexity is stated in terms of n, ℓ, and k.

f) Assume your method showed that exactly K people can make it to the conference, find the smallest p such that $K + 1$ people are now able to make it. Argue the correctness and the time complexity of your method.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no h[illegible]).
- Start by typi[illegible]) at the top of the first page of your submission. Do **not** type[illegible]
- Submit only [illegible]he questions. Do **not** copy the questions.
- When asked [illegible]nglish description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, when designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you marks for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.

Here is template LaTeX CODE for drawing the residual graph (especially for those unfamiliar with tikz)

```
\begin{figure}[htbp]
\begin{center}
\vspace{1ex}
\tikzstyle{arc}[illegible]dth=1pt]
\tikzstyle{vert[illegible],very thick,inner sep=2pt]
\begin{tikzpict[illegible]cale=1]
\node[vertex,la[illegible]) at (0,0) { };
\node[vertex,label=above:$a$] (a) at (4,2) { };
\node[vertex,label=below:$b$] (b) at (4,-2) { };
\node[vertex,label=above:$c$] (c) at (8,4) { };
\node[vertex,label=above right:$d$] (d) at (8,0) { };
\node[vertex,label=below:$e$] (e) at (8,-4) { };
\node[vertex,label=above:$f$] (f) at (12,2) { };
\node[vertex,label=below:$g$] (g) at (12,-2) { };
\node[vertex,label=right:$t$] (t) at (16,0) { };

\draw[arc] (s) to[out=60,in=180]  node [red, fill=white] {$0$} (a);
\draw[arc] (a) to  node [red, fill=white] {$1$} (s);

\draw[arc] (s) to[out=270,in=180] node [red, fill=white] {$2$} (b);
\draw[arc] (b) to node [red, fill=white] {$3$} (s);

\draw[arc] (a) to[out=60,in=180] node [red, fill=white] {$4$} (c);
\draw[arc] (c) to node [red, fill=white] {$5$} (a);

\draw[arc] (a) to[out=270,in=180] node [red, fill=white] {$6$} (d);
\draw[arc] (d) to node [red, fill=white] {$7$} (a);

\draw[arc] (b)[out=60,in=180] to node [red, fill=white] {$8$} (d);
\draw[arc] (d) to node [red, fill=white] {$9$} (b);

\draw[arc] (b)[out=270,in=180] to node [red, fill=white] {$10$} (e);
\draw[arc] (e) to node [red, fill=white] {$11$} (b);

\draw[arc] (c)[out=0,in=135] to node [red, fill=white] {$12$} (f);
\draw[arc] (f) to node [red, fill=white] {$13$} (c);

\draw[arc] (d)[out=60,in=300] to node [red, fill=white] {$14$} (c);
\draw[arc] (c)[out=240,in=135] to node [red, fill=white] {$15$} (d);

\draw[arc] (d)[out=240,in=135] to node [red, fill=white] {$16$} (e);
\draw[arc] (e)[out=60,in=300] to node [red, fill=white] {$17$} (d);

\draw[arc] (d)[out=0,in=135] to node [red, fill=white] {$18$} (g);
\draw[arc] (g) to node [red, fill=white] {$19$} (d);
```

```
\draw[arc] (e)[out=0,in=240] to node [red, fill=white] {$20$} (g);
\draw[arc] (g) to node [red, fill=white] {$21$} (e);

\draw[arc] (f)[ node [red, fill=white] {$22$} (t);
\draw[arc] (t) ll=white] {$23$} (f);

\draw[arc] (f)[ to node [red, fill=white] {$24$} (g);
\draw[arc] (g)[ o node [red, fill=white] {$25$} (f);

\draw[arc] (g)[out=0,in=240] to node [red, fill=white] {$26$} (t);
\draw[arc] (t) to node [red, fill=white] {$27$} (g);
\end{tikzpicture}
\end{center}
\caption{Residual Graph}
\end{figure}
```


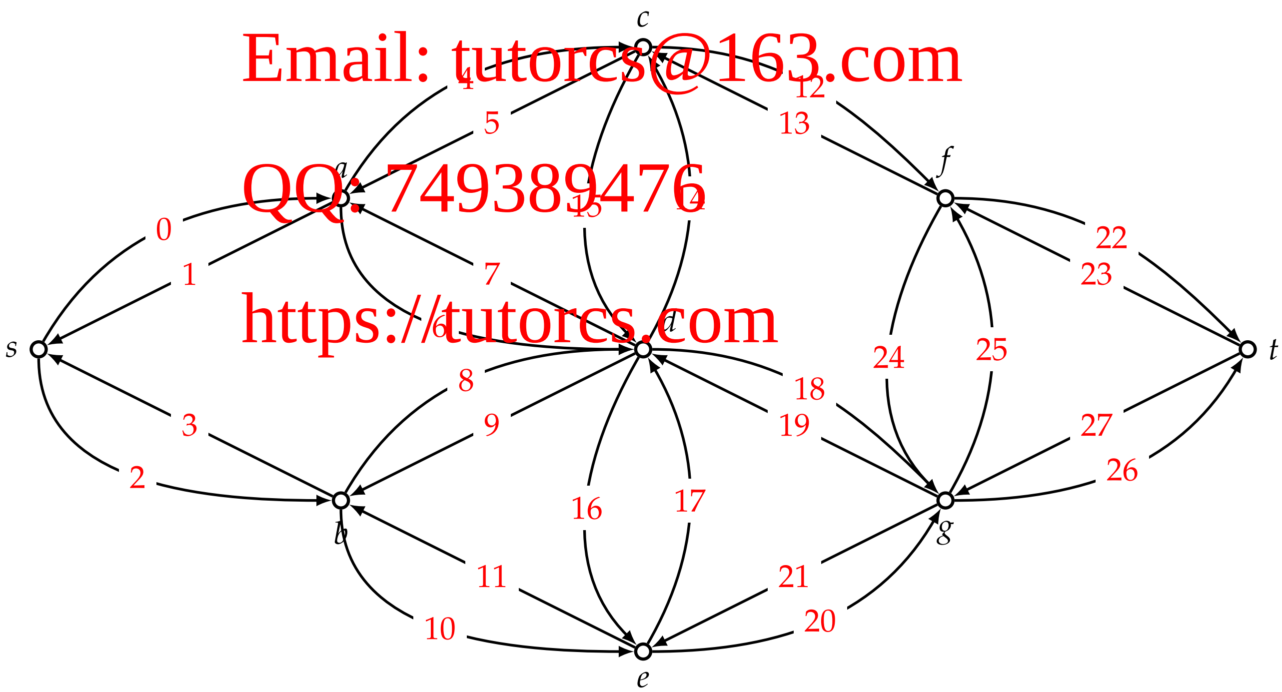

Figure 5: Residual Graph