Network Algorithms: Exercise 1
Asymptotic Notations and Graph Terminologies

Dr. Stefan Schmid, Arne Ludwig, Srivatsan Ravi, Lalith Suresh

NOTE: Exercise 1 will be a bonus worksheet.

(1) In this question, we recall the definitions of the asymptotic notations.

Let f, g: $\mathbb{N} \to \mathbb{N}$.

We say f is $O(g)$ if $\exists c \in \mathbb{N}$ and n_0 such that $\forall n \geq n_0$, $f(n) \leq c \cdot g(n)$.

We say f is $o(g)$ if $\forall c \in \mathbb{N}$, there is n_0 such that $\forall n \geq n_0$, $f(n) \leq c \cdot g(n)$ i.e. f grows strictly more slowly than *any* arbitarily small positive constant multiple of g.

We say f is $\Omega(g)$ if g is $O(f)$.

We say that f is $\Theta(g)$ if f is both $O(g)$ and $\Omega(g)$.

Now answer the following questions and explain your solutions in detail.

(1a) Let $f_1(n) = 23n^2$. Is $f_1(n) \in o(n^3)$?

(1b) Let $f_2(n) = 42n^3 + n^2 \log n$. Is $f_2(n) \in \Theta(n^3)$?

(1c) Is $3^{\log n} \in O(n)$?

(1d) Consider the geometric summation $\sum_{i=0}^{n} \frac{1}{2^i}$. Is it $O(1)$?

(1e) Consider the problem of sorting a sequence of n distinct elements from a set S such that given any two elements in S, we can compare them (for example, you can think of S to be a finite set of natural numbers). This is the only operation available to gain information about the sequence. Show that any algorithm that uses sorts by comparisons on some sequence of length n must use at least $c \cdot n \log n$ comparisons for some constant c.
Hint: $n! \geq (\frac{n}{2})^{\frac{n}{2}}$

(2) Given the following topology:

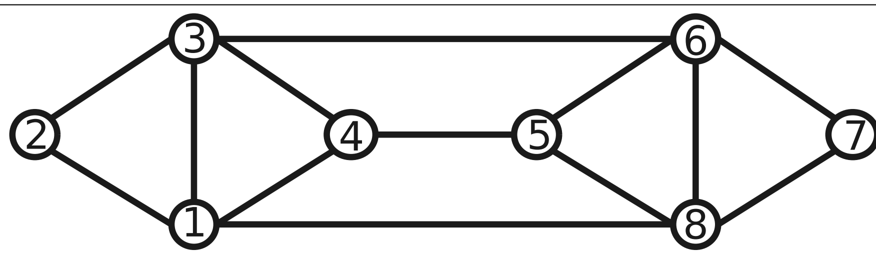

Figure 1: Graph G

Provide the following graph properties regarding G:

(2a) Maximum degree

(2b) Diameter (Definition 3.2 in the Lecture Notes)

(2c) Radius (Definition 3.2 in the Lecture Notes)

(2d) A minimal maximal matching (Definition 5.15 in the Lecture Notes)

(2e) A perfect matching (Definition 5.15 in the Lecture Notes)

(2f) Chromatic number for the coloring of G (Definition 4.3 in the Lecture Notes)

(2g) A maximum independent set (Definition 5.1 in the Lecture Notes)

(3) Recall the pledge algorithm from the first lecture. Explain under which circumstances this algorithm does not work.

(4) Recall Amdahl's law from the first lecture. Assume a program needs 25 hours to run on a machine with a single CPU. Three hours are needed for initilization and can not be parallelized. The rest of the work can be parallelized on equally powerful CPUs.

(4a) What is the minimal amount of CPUs in order to achieve a speedup of at least 3?

(4b) What is the theoretical maximum speedup for this program?