1 Big-O notation

For each of the following, indicate whether \( f(n) = O(g(n)) \), \( f(n) = \Omega(g(n)) \), or \( f(n) = \Theta(g(n)) \). Justify your answer.

- \( f(n) = 100 \cdot n \) and \( g(n) = n^{1.1} \);
- \( f(n) = n^3 + 2n^2 + 10 \) and \( g(n) = (\log_2 n)^5 \);
- \( f(n) = 2^n \) and \( g(n) = \sqrt{n} \sqrt{n} \);
- \( f(n) = n^{\log_2 \log_2 n} \) and \( g(n) = 2 \cdot (\log_2 n)^{\log_2 n} \).

1.1 \( f(n) = 100 \cdot n \) and \( g(n) = n^{1.1} \)

This is easiest to approach by definition. There exists \( c = 100 \) and \( n_0 = 1 \) such that \( \forall n \geq n_0 \) we have \( 100 \cdot n \leq 100 \cdot n^{1.1} \). Hence \( f(n) = O(g(n)) \).

On the other hand, \( f(n) \neq \Omega(g(n)) \). Assume, by contradiction, that for some value \( n_0 \) and \( c \) we have \( f(n) \geq c \cdot g(n) \) for all \( n \geq n_0 \). This can not happen, because for any \( c \) exists \( n_1 \) such that \( c \cdot n_1^{0.1} > 1 \), hence, for all \( n \geq n_1 \) we have \( f(n) < c \cdot g(n) \). Hence also \( f(n) \neq \Theta(g(n)) \).

1.2 \( f(n) = n^3 + 2n^2 + 10 \) and \( g(n) = (\log_2 n)^5 \)

The answer \( f(n) = \Omega(g(n)) \) is easiest to see by computing the limit with L’Hospital rule consecutively because of the indetermination \( \frac{\infty}{\infty} \).

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{n^3 + 2n^2 + 10}{(\log_2 n)^5} = \lim_{n \to \infty} \frac{(n^3 + 2n^2 + 10)'}{(\log_2 n)^5'} = \lim_{n \to \infty} \frac{3n^2 + 4n^2}{5(\log_2 n)^4} = \lim_{n \to \infty} \frac{3n^2 + 4n^2}{5(\log_2 n)^4} =
\]

\[
\frac{\ln 2}{5} \lim_{n \to \infty} \frac{(\log_2 n)^3}{(\log_2 n)^3} = \frac{\ln 2}{5} \lim_{n \to \infty} \frac{(27n^3 + 16n^2)'}{(\log_2 n)^4} = \frac{\ln 2}{120} \lim_{n \to \infty} \frac{(81n^3 + 32n^2)'}{(\log_2 n)^4} =\]

\[
\frac{(\ln 2)^2}{20} \lim_{n \to \infty} \frac{(9n^3 + 8n^2)'}{(\log_2 n)^3} = \frac{(\ln 2)^3}{60} \lim_{n \to \infty} \frac{(243n^3 + 64n^2)'}{(\log_2 n)^2} = \frac{(\ln 2)^4}{120} \lim_{n \to \infty} \frac{1}{(\log_2 n)^2} = \infty
\]
1.3 \( f(n) = 2^n \) and \( g(n) = \sqrt{n^\sqrt{n}} \)

The answer is again \( f(n) = \Omega(g(n)) \).

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{2^n}{\sqrt{n^\sqrt{n}}} = \lim_{n \to \infty} \frac{2^n}{\left(\sqrt{n}\right)^\sqrt{n}} = \infty
\]

Where we need to find \( \lim_{n \to \infty} \frac{2\sqrt{n}}{\sqrt{n}} = \infty \) for the last claim. This can be easily seen with an exchange of variables with \( x = \sqrt{n} \) where \( x \to \infty \) as \( n \to \infty \).

Finally, we can apply the L’Hospital rule:

\[
\lim_{x \to \infty} \frac{2^x}{x} = \lim_{x \to \infty} \frac{2^x \ln 2}{1} = \infty
\]

1.4 \( f(n) = n^{\log_2 \log_2 n} \) and \( g(n) = 2 \cdot (\log_2 n)^{\log_2 n} \)

In this case \( f(n) = \Theta(g(n)) \) as both definitions for big-O and \( \Omega \) are satisfied by \( c = \frac{1}{2} \) for any \( n_0 \). This is true because the functions are actually equal up to constant factor 2.

\[
n^{\log_2 \log_2 n} = (2^{\log_2 n})^{\log_2 \log_2 n} = 2^{\log_2 n \cdot \log_2 \log_2 n} = (2^{\log_2 \log_2 n})^{\log_2 n} = (\log_2 n)^{\log_2 n}
\]

2 Two spanning trees

Let \( T_1(V, E_1) \) and \( T_2(V, E_2) \) be two spanning trees of the (undirected, connected, finite) graph \( G(V, E) \). Prove that for every edge \( e \in E_1 \setminus E_2 \) there exists an edge \( e' \in E_2 \setminus E_1 \) such that each of the edge sets \( (E_1 \cup \{e'\}) \setminus \{e\} \) and \( (E_2 \cup \{e\}) \setminus \{e'\} \) defines a spanning tree.

Firstly, we can make some observations. For example \(|E_1| = |E_2| = |V| - 1\) because they are trees of the same vertices. Hence also \(|(E_1 \cup \{e'\}) \setminus \{e\}| = |(E_2 \cup \{e\}) \setminus \{e'\}| = |V| - 1\). Hence, by definitions these are spanning trees if they are connected. In addition, clearly \( e \neq e' \).

By definition \((E_1 \setminus \{e\})\) has \(|V| - 2\) edges and is disconnected. Assume that \( e = \{a, b\} \) for \( a, b \in V \). Then we can consider the two sets of disconnected vertices in \( A \subset V \) and \( A \subset V \) such that \( a \in B \) and vertices in \( A \) or \( B \) are connected and there is no edge between \( A \) or \( B \).

Consider a circuit \( C \) in \( E_2 \cup \{e\} \). By definition, there exists an edge \( p \in C \) such that \( p = \{a', b'\} \neq e \) for \( a' \in A \) and \( b' \in B \) because the circuit contains vertices from both \( A \) and \( B \), hence there are at least two such edges (and only one of them is \( e \)). By definition also \( p \neq e \) and \( p \notin E_1 \) as \( E_1 \setminus \{e\} \) has no edge between \( A \) and \( B \). Now, consider \((E_2 \setminus \{e\}) \cup p\) which is connected because \( p \) connects \( A \) and \( B \). In addition, it has \(|V| - 1\) edges, therefore it is a spanning tree. Differently, consider \((E_2 \setminus \{p\}) \cup \{e\}\). We know that \( e \) and \( p \) belong to one circuit in \( T_2 \) so \((E_2 \setminus \{p\}) \cup \{e\}\) is also connected and has \(|V| - 1\) edges. Therefore it is also a spanning tree. This is a contradiction as there exists such \( e' = p \).
3 Analogous weight functions

Let \( G(V, E) \) be an undirected, connected, finite graph. Let \( w : E \to \mathbb{R}^+ \) and \( w' : E \to \mathbb{R}^+ \) be two weight functions, such that

\[
\forall e_1, e_2 \in E : w(e_1) \leq w(e_2) \iff w'(e_1) \leq w'(e_2) .
\]

Prove that \( T \) is a minimum spanning tree of \( G \) with respect to \( w \) if and only if \( T \) is a minimum spanning tree of \( G \) with respect to \( w' \).

The following presents two ideas for solving this problem.

3.1 Idea 1: Analysing the edges in the tree

Assume by contradiction that there is a minimum spanning tree \( T_1(V, E_1) \) of \( G \) with respect to \( w \) that is not minimum with respect to \( w' \). Let \( T_2(V, E_2) \) where \( E_1 \neq E_2 \) be some minimum spanning tree of \( G \) with respect to \( w' \).

\( T_1 \) and \( T_2 \) are both spanning trees of \( G \), hence we can apply the result of Exercise 1. Hence, for each \( e \in E_1 \setminus E_2 \) there exist \( e' \in E_2 \setminus E_1 \) such that sets of edges \( E_1' = (E_1 \cup \{e\}) \setminus \{e\} \) and \( E_2' = (E_2 \cup \{e\}) \setminus \{e'\} \) also define spanning trees. We use the extended notation of the weight function as

\[
w(E) = \sum_{e \in E} w(e) .
\]

We have three possibilities:

1. \( w(e) < w(e') \) and \( w'(e) < w'(e') \). Then \( w'(E_2') = w'(E_2) + w'(e) - w'(e') < w'(E_2') \) which is a contradiction because \( T(V, E_2) \) was a minimum spanning tree with respect to \( w' \).

2. \( w(e') < w(e) \) and \( w'(e') < w'(e) \). Then \( w(E_1) = w(E_1') + w(e') - w(e) < w(E_1) \) which is a contradiction because \( T(V, E_1) \) was a minimum spanning tree with respect to \( w \).

3. \( w(e) = w(e') \) and \( w'(e) = w'(e') \). In this case we can define \( T'_2(V, E'_2) \) as a new minimum spanning tree of \( G \) with respect to \( w' \) because \( w'(E_2) = w'(E'_2) \). Now, there are two possibilities:

   (a) \( E'_2 = E_1 \) then we have \( T_1 = T'_2 \) which is a contradiction with the initial assumption.

   (b) \( E'_2 \neq E_1 \) then we start this analysis again with the same \( T_1 \), but use \( T'_2 \) instead of \( T_2 \). Either the edge \( e \) that we pick will give a contradiction for the conditions 1, 2 or 3a or we can do another change for the three \( T'_2 \) that we substitute for \( T'_2 \) and so on.

The following claim clarifies why this always terminates and at some point we end the process in one of the contradicting branches.

Claim: The process of substituting \( T_2 \) with \( T'_2 \) in previous branch 3b will terminate with at most \(|V| - 1\) steps.

Reasoning: Initially we have \( E_1 \cap E_2 = R \) and for each \( e \) that we pick in the analysis \( e \notin R \) because \( e \notin E_2 \) and \( e' \notin R \) because \( e' \notin E_1 \). However, for \( E_1 \cap E_2 = R \cup \{e\} \). The maximum
possible size of $R$ is clearly $\max(|R|) = |E_1| = |E_2| = |V| - 1$. Each iteration will increase the size of $R$ exactly by one, therefore there will be a step where we add some $p$ so that $R = E_1$ and therefore $T_2' = T_1$ as in branch 3a. □

Therefore, if $T$ is a minimum spanning tree for $G$ with respect to $w$ then it is also a minimum spanning tree with respect to $w'$.

The other direction changes the roles of $w$ and $w'$ in the previous analysis, but is otherwise the same.

Therefore, $T$ is a minimum spanning tree of $G$ with respect to $w'$ if and only if it is a minimum spanning tree with respect to $w'$.

### 3.2 Idea 2: Kruskal algorithm

For every minimum spanning tree $T$ of $G$ there exists a run of Kruskal algorithm that finds this $T$. (Note that we did not prove this in class so that requires a separate proof.) These runs are defined by the different order of sorting.

Let $T$ be a minimum spanning tree of $G$ with respect to $w$. Now, consider a run of the Kruskal algorithm that finds $T$. Without lessening of generality assume that the initial sorting yields $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_{|E|})$ where $i < j$ implies $w(e_i) \leq w(e_j)$. Starting Kruskal algorithm with edges ordered as $[e_1, e_2, e_3, \ldots, e_{|E|}]$ yields $T$. However, by definition this is also a valid ordering for $w'$ because $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_{|E|})$ implies $w'(e_1) \leq w'(e_2) \leq \ldots \leq w'(e_{|E|})$. The rest of the algorithm is independent of the weight function, therefore it would give the same $T$ for $w$ and $w'$ because we can use the same sorter order. By correctness of the Kruskal algorithm we know that $T$ is therefore a minimum spanning tree for both of these cases.

The other direction where $T$ being a MST for $w'$ implies that it is also a spanning tree for $w$ is analogous.

### 4 Power of the Kruskal algorithm

Let $G(V,E)$ be an undirected, connected, finite graph with weight function $w : E \rightarrow \mathbb{R}^+$. Let $T$ be a minimum spanning tree of $G$. Show that there exists a run of Kruskals algorithm that finds $T$ (for suitable ordering of edges).

Assume, by contradiction, that there exists a minimum spanning tree $T_1(V,E_1)$ of $G$ such that no run of the Kruskal algorithm finds this tree. Consider also a run of Kruskal algorithm that finds the spanning tree $T_2(V,E_2)$. For simplicity (and without lessening of generality), assume that the weight function is such that $w(e_i) \leq w(e_j)$ if $i < j$ and the sorting order that produces $T_2$ is by the order of the indices.

Assume that $e_1$ is the first such edge that differs between $E_1$ and $E_2$. Hence, it is the first edge where the algorithm makes a different choice for $T_2$ than $T_1$. There are two cases:

1. Assume, that $e_1$ is added to $E_2$ by the algorithm, but that it does not belong to $E_1$. The set $E_1 \cup \{e_1\}$ would contain a circuit. From the fact that $e_1$ is chosen to $E_2$ we know that it does
not form a cycle with only elements with the smaller index than \( i \) as there is no circuit in \( E_2 \). Hence the circuit in \( E_1 \cup \{ e_i \} \) must contain at least one edge with a larger index than \( i \). Therefore, there exists \( e_j \) where \( j > i \) and \( e_j \in E_1 \) but \( e_j \notin E_2 \) that is on the circuit. In addition, assume that \( e_j \) is the edge with the least weight among the suitable edges on the circuit.

There are two cases:

(a) \( w(e_i) = w(e_j) \). In such case we could consider a different run of the Kruskal algorithm where we swap the places of \( e_i \) and \( e_j \) in the sorting. This algorithm would produce \( T_3(V, E_3) \) with the condition that all edges \( e_k \) with \( k < i \) and \( e_j \) are either in both minimum spanning trees \( T_3 \) and \( T_1 \) or in neither. Clearly, using \( e_j \) instead of \( e_i \) would not introduce a circuit because there is no circuit in \( E_1 \), hence it is chosen to \( E_3 \) by the algorithm.

We could continue the analysis by finding the first edge that differs between \( T_3 \) and \( T_1 \) if \( E_1 \neq E_3 \). However, if \( E_1 = E_3 \) then we have found a run of Kruskal algorithm that finds \( T_1 \), which is a contradiction with the initial assumption.

(b) \( w(e_i) < w(e_j) \). We could consider a graph \( T_3(V, (E_1 \cup \{ e_i \}) \setminus \{ e_j \}) \). This is a spanning tree because it has \( |V| - 1 \) edges and it is connected because \( e_i \) and \( e_j \) formed a circuit in \( E_1 \). The weight of \( T_3 \), however, is less than \( T_1 \) because \( w(e_i) < w(e_j) \) which is a contradiction because \( T_1 \) was a minimum spanning tree. Therefore, this case can not happen.

2. Assume that \( e_i \) is a part of \( E_1 \), but is not chosen to \( E_2 \) by the Kruskal algorithm. The fact that \( e_i \) is not chosen to \( E_2 \) means that there is a circuit with \( e_i \) and edges previously added to \( E_2 \). However, the previous edges of \( E_2 \) are also part of \( E_1 \) and the fact that \( e_i \in E_1 \) contradicts the possibility of a circuit. Hence, it is not possible to find such an edge and this case can not happen.

Hence, there always exists a run of Kruskal algorithm that finds a minimum spanning tree \( T \). The main summary of the proof is that always when the Kruskal algorithm finds a different spanning tree then the difference is in the edges with the same weight and we can reorder them in the initial sorting to produce the necessary tree.

5 Uniqueness of the minimum spanning tree

Let \( G(V, E) \) be an undirected, connected, finite graph with weight function \( w : E \rightarrow \mathbb{R}^+ \). It is known that the weights of the edges in \( E \) are all different. Show that \( G \) has a unique minimum spanning tree.

Assume by contradiction, that there are two minimum spanning trees \( T_1(V, E_1) \) and \( T_2(V, E_2) \). We use the extended notation of the weight as

\[
w(E) = \sum_{e \in E} w(e).
\]
By definition $w(E_1) = w(E_2)$ and is minimal possible for a spanning tree of $G$. We also know that the trees are different, hence $E_1 \neq E_2$. Therefore we can use the proposition of Exercise 2 to find the edges $e \in E_1 \setminus E_2$ and $e' \in E_1 \setminus E_1$ as defined there $e \neq e'$.

Hence, set of edges $E'_1 = (E_1 \cup \{e'\}) \setminus \{e\}$ and $E'_2 = (E_2 \cup \{e\}) \setminus \{e'\}$ also define a spanning tree. Now, we can compute the weight of these trees as follows.

$$w(E'_1) = w(E_1) - w(e) + w(e')$$
$$w(E'_2) = w(E_2) - w(e') + w(e)$$

From the precondition of the theorem we know that $w(e) \neq w(e')$. Therefore we have two cases:

1. if $w(e) < w(e')$ then $w(E'_2) = w(E_2) - w(e') + w(e) < w(E_2)$ which is a contradiction, because then $T(V, E'_2)$ would be a spanning tree with smaller weight than the minimum spanning tree of $G$.
2. if $w(e') < w(e)$, then $w(E'_1) = w(E_1) - w(e) + w(e') < w(E_1)$, which is again a contradiction because $T_1(V, E_1)$ is a minimum spanning tree of $G$.

Therefore, it is impossible for any two different minimum spanning trees to exist when all the edges have different weights.