# Chapter 3. Section 1

## Minimum-Weight Spanning Trees

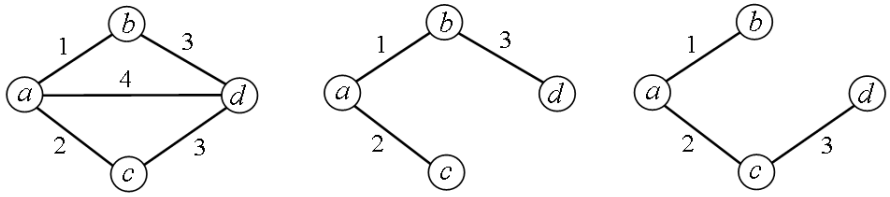
Suppose $G = (V, E)$ is a connected graph. Then a maximal subtree of $G$ will automatically be maximum and will reach every vertex $v \in V$. Such a subtree is said to be **spanning**.

Another viewpoint: a spanning tree is a minimum subset of edges that preserves connectedness of $G$.

**Exercise:** a subgraph of $G$ that has fewer vertices and/or edges than $G$ is called **proper** . Prove that a connected graph that has a cycle contains a proper connected subgraph. Hence prove that a connected subgraph of $G$ with fewest edges is a spanning tree.

We now add a cost or **weight** to each edge and ask for a spanning tree of minimum total edge weight.

**Example:** in the graph below left the numbers on the edges represent weights. Either of the subgraphs on the right is a minimum-weight spanning tree for these weights.



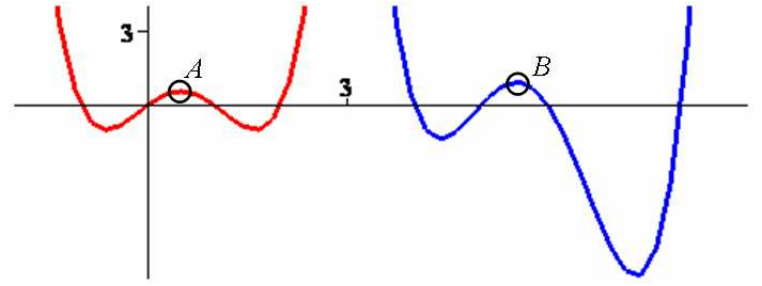

**Note:** remember that a drawing of a graph is not the same as the mathematical object. So adding numbers on edges is not mathematics either, it is just a convenient pictorial representation. The graph above would properly be described as a graph $G = (V, E)$, together with a weight function $w : E \to \mathbb{R}$:

$$\begin{aligned} V &= \{a, b, c, d\} \\ E &= \{ab, ac, ad, bd, cd\} \\ & \quad w(ab) = 1,\ w(ac) = 2,\ w(ad) = 4,\ w(bd) = 3,\ w(cd) = 3. \end{aligned}$$

## Greedy Algorithms

A feature of the Maximal_Subtree algorithm is that we never worry whether adding a particular edge might prevent progress in the future: if we see a valid edge, we automatically add it to the tree. This is referred to as optimising 'greedily': optimal current action incurs no future penalty.

In traditional optimisation terms we can illustrate this with the two functions plotted below:




At point $A$, a hill-climbing algorithm minimising the function can descend right or left: either choice will achieve the same minimum value[1]. At point $B$, however, descending to right or left looks equally promising. But eventually the left-hand descent proves to be non-optimal: it takes us to a minimal point which is not minimum. We conclude that polynomials cannot be minimised greedily by hill-climbing.

[1]The left-hand function is $(x + 1)\,x\,(x - 1)\,(x - 2)$ which has minima at $(-1/\phi, -1)$ and $(\phi, -1)$, where $\phi$ is the golden ratio. The right-hand function is $(x - 4)\,(x - 5)\,(x - 6)\,(x - 8)$, with minima at approx. $(4.4, -1.4)$ and $(7.3, -6.9)$

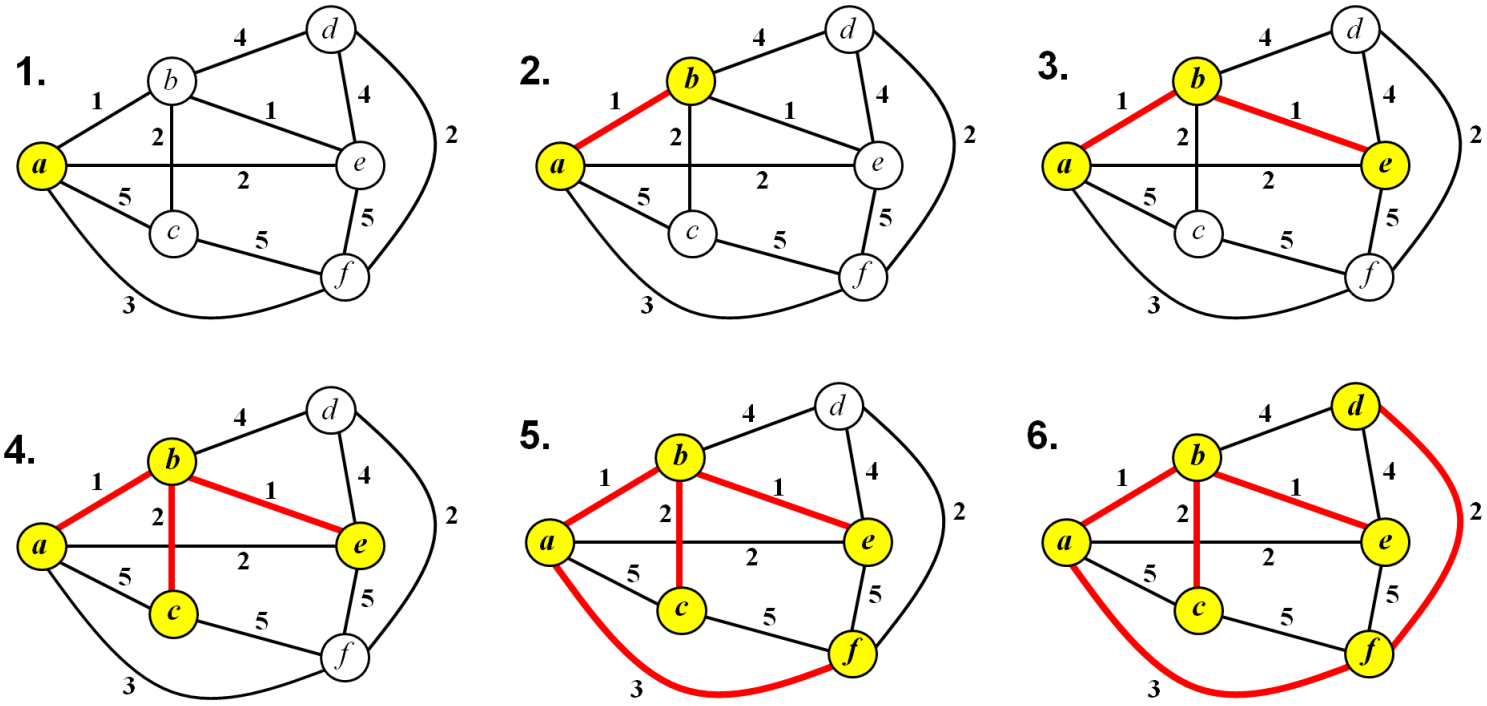
# The MWST Algorithm

Not many optimisation problems can be solved greedily. However, finding a minimum-weight spanning tree can, just by adding the words "of minimum weight" in one line of the Maximal_Subtree algorithm! We will call this algorithm MWST:

Algorithm **MWST**
**Input:** graph $G = (V, E)$
**Output:** spanning tree $T = (V(T), E(T))$ of $G$ of minimum weight
1. choose any vertex $v$
2. $V(T) := \{v\}$, $E(T) := \emptyset$ # initialise the subtree to be the single vertex $v$ and no edges
3. **while** some edge leaves the tree **do**
4. choose a leaving edge $xy$ (with $x \in V(T)$, $y \notin V(T)$) of minimum weight
5. $V(T) := V(T) \cup \{y\}$ # add the new vertex $v$ to $T$
6. $E(T) := E(T) \cup \{xy\}$ # add the new edge $xy$ to $T$

**return**$(T)$
**end**

**Example:** of Minimum-Weight_Spanning_Tree in action.




We now have our usual obligations: we must prove that MWST is correct and we must analyse its complexity.

We take the latter first:

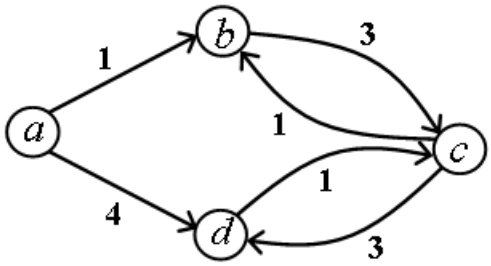
**Complexity:** MWST has the same complexity as Minimal_Subtree, i.e. $O(|V|^2|E|)$. We have the additional requirement to find a lowest weight leaving edge at step 4. but our analysis of Minimal_Subtree assumed we were checking all edges of $G$ not already in $T$, so we can check the weights of the edges at the same time.

In the next lecture we will give a variant of MWST called Prim's Algorithm which takes $O(|V|^2 + |E|)$ steps. This is much faster even if $|E|$ is small (meaning about the same as the number of vertices: $|E| = O(|V|)$.

Now we prove **correctness**. We remark immediately that it is not obvious that we can find minimum-weight spanning trees greedily. To emphasise this point, suppose we try to create a digraph version

of MWST which finds a minimum weight spanning tree *directed away from a given vertex v* (this is called at **out-tree** at *v*.

**Example:** the greedy approach to building an out-tree from the left-hand vertex in the digraph below gives the out-tree of total weight 7 whose edges are: *ab*, *bc*, *cd* (remember these are directed edges, so *bc* and *cb* are different edges). However, a minimum-weight out-tree has weight 6: *ad*, *dc*, *cb*. There is no way we can start building an out-tree greedily with edge *ad*: it requires future planning to do that!




So we must give a proof that everything works OK for the undirected case. First some terminology: we use $w(e)$ to denote the weight of edge $e$. If $X \subset E$ then $w(X)$ will denote the total edge weight of all edges in $X$. In symbols: $w(X) = \sum_{e \in X} w(e)$. If $T$ is a subtree of $G$ we will say that $T$ is **extendable** if $T$ is contained in (meaning, is a subtree of) some minimum-weight spanning tree of $G$.

Now our proof rests on the following:

**Lemma** If $T$ is an extendable subtree of $G$ and $e$ is has minimum-weight among the edges leaving $T$ then $T + e$ is again extendable.

**Remark:** for a spanning tree $T$, extendable is equivalent to minimum-weight. So when the MWST algorithm has completed its spanning tree the Lemma guarantees this will be a minimum-weight spanning tree.

**Proof of Lemma:** $T$ is extendable so $T$ is contained in a minimum-weight spanning tree, say $\hat{T}$. If $e$ is an edge of $\hat{T}$ then $T + e$ is still contained in $\hat{T}$ and is therefore extendable as required.

So suppose that $e \notin \hat{T}$. Then $\hat{T} + e$ contains a cycle (since now it has too many edges to be a tree). This cycle must contain $e$, so it must reenter $T$, via edge, say $f$. Since edges are undirected $f$ is also a leaving edge for $T$ so $w(f) \geq w(e)$ since $e$ was a minimum-weight leaving edge for $T$.

Now $\hat{T} + e - f$ is again a tree, since deleting $f$ breaks the cycle, and

$$w(\hat{T} + e - f) = w(\hat{T}) + w(e) - w(f) \leq w(\hat{T}).$$

Since $\hat{T}$ was a minimum-weight spanning tree so must $\hat{T} + e - f$ be. And $\hat{T} + e - f$ contains $T + e$. So $T + e$ is extendable. □

# Maximum-Weight Spanning Tree

We may wish to maximise rather than minimise total spanning tree edge weight. Simple — we just replace the weight $w(e)$ of edge $e$ with $-w(e)$. Now the minimising total edge weight will maximise the total in the original graph.

**Or:** replace the words ‘minimum weight’ in line 4. of the MWST algorithm with the words ‘maximum weight’. However, this might mean changing the code of a large complex computer program, so adjusting the input weights is usually preferable!