Solutions to Homework 5: NP-Completeness and Approximations

Solution 1: We need to prove that (i) SIS $\in$ NP and (ii) SIS is NP-hard. For the latter, we will show that IS $\leq_P$ SIS.

- SIS $\in$ NP: Given an instance (G, k) for SIS, where $G = (V, E)$, the certificate consists of a subset $V' \subseteq V$ of size k. In polynomial time we can compute the distances between all pairs of vertices (e.g., using the Floyd-Warshall algorithm) and then check for each pair $u, v \in V'$ that the distance between them is 3 or greater. If so, the verification accepts and otherwise it rejects.

- IS $\leq_P$ SIS: Consider any instance (G, k) for IS. Our objective is to compute an instance (G', k') for SIS such that G has an independent set of size k if and only if G' has a strong independent set of size k'. Intuitively, we need to translate the notion of two vertices being nonadjacent to the notion that two vertices are at distance three or higher. This suggests the idea of inserting a vertex into the middle of each edge, called a *mid-edge vertex*, thus doubling distances in the graph. Unfortunately, this does not work, since there is nothing forbidding us from placing these mid-edge vertices into the independent set.

  To deal with this issue, we will render it useless to include these vertices by connecting them all to each other. At first, this might seem like a bad thing to do, since vertices that were far apart in distance are suddenly much closer. This essentially causes every vertex in the original graph to be at distance at most two from every other original vertex. As we shall see, this is not a problem. The reduction is illustrated in Fig. 1.

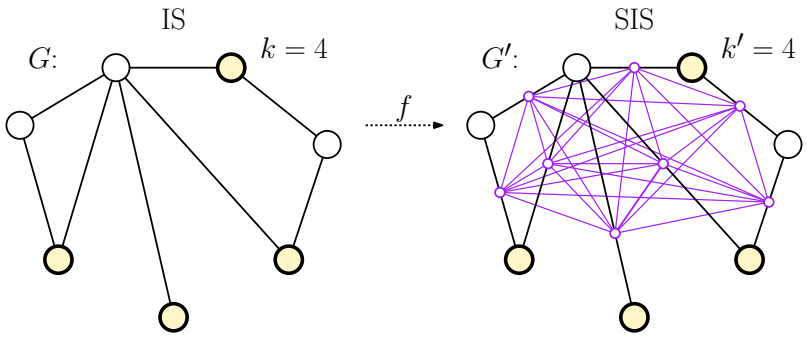

Figure 1: Reduction from IS to SIS.

  More formally, given a graph $G = (V, E)$ and integer k for IS, we produce a new graph $G' = (V', E')$ and integer k' as follows. First set $k' \leftarrow k$. Next, for each edge $(u, v) \in E$, we create a new mid-edge vertex w_{uv}, and we connect all these mid-edge vertices together in a completely connected subgraph. Next, we replace each edge $(u, v) \in E$, the two edges (u, w_{uv}) and (v, w_{uv}). Let G' denote the resulting graph (see Fig. 1). We output (G', k').

  Letting $n = |V|$ and $m = |E|$, G' has $n + m$ vertices and $O(2m + m^2) = O(m^2)$ edges. The construction can be performed in time proportional to the size of G', which is $O(n + m^2)$, which is polynomial in the input size. Correctness is established in the following claim.

Claim: G has an independent set of size k if and only if G' has an strong independent set of size $k' = k$.

Proof: It will simplify the proof to assume that $k \geq 2$, and that G has no isolated vertices. (If $k = 1$, then the answer to both problems is trivially "yes" for any graph. Isolated vertices can always be added to any independent set or strong independent set. So, we can remove them and adjust the value of k accordingly.)

($\Rightarrow$) If G has an independent set V' of size k, we assert that the same vertices form a strong independent set in G'. If $u, v \in V'$, then they are not adjacent in G, meaning that we need at least two edges to get from u to v in G. In G', the shortest distance between them is at least three (from u to a mid-edge vertex for an edge incident to u, then to another mid-edge vertex for an edge incident to v, then to v itself). Therefore G' has a strong independent set of size $k' = k$.

($\Leftarrow$) Conversely, suppose that G' has an strong independent V' set of size k', where $k' \geq 2$. We first assert that we can assume that all the vertices of V' are taken from the original set V, and do not include any mid-edge vertices. Observe that, because we assume there are no isolated vertices, every mid-edge vertex is within distance two of every vertex in G' (from the mid-edge vertex to any other mid-edge vertex, then to an original vertex incident to the associated edge). Thus, any strong independent set containing a single mid-edge vertex has size at most one, and by hypothesis $k' \geq 2$. If two of the original vertices are in V', then they must be separated by a distance of at least three. This implies that they could not have been adjacent in the original graph G (for otherwise, they would be distance two, going through the mid-edge vertex of their shared edge). Therefore, V' is an independent set in G of size $k = k'$.

Solution 2:

(a) Given a directed graph $G = (V, E)$, the reduction takes any vertex u and replaces it with two new vertices, u' and u''. All the edges outgoing from the original u now are outgoing from u' (by replacing each edge (u, v) with (u', v)), and all the edges incoming to the original u now are incoming to u'' (by replacing each edge (v, u) with (v, u'')). Let G' be the resulting graph (see Fig. 2(a)). Correctness is established by the following claim.

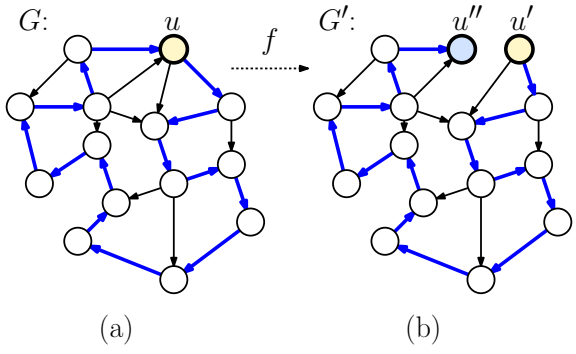

Figure 2: DHC to DHP reduction.

Claim: G has a Hamiltonian cycle if and only if G' has a Hamiltonian path.

Proof: ($\Rightarrow$) Suppose that G has a Hamiltonian Cycle $\langle u_0, u_1, \ldots, u_{n-1}\rangle$. Since we can start the cycle wherever we like, we may assume that $u_0 = u$ in our construction. It follows that G' has the path $\langle u', u_1, \ldots, u_{n-1}, u''\rangle$. Clearly, this is a Hamiltonian path in G' (see Fig. 2(b)).

($\Leftarrow$) Suppose that G' has a Hamiltonian Path. The path must start at u' and end at u'', because these vertices have only outgoing and incoming edges, respectively. Therefore, the path must have the form $\langle u', u_1, \ldots, u_{n-1}, u''\rangle$, for some sequence $u_1, \ldots, u_{n-1}$ that forms a simple path in G'. We can convert this into a cycle in G by replacing u' and u'' with the single node u. Therefore, G has a Hamiltonian Cycle.

(b) Given a directed graph $G = (V, E)$, the reduction replaces each vertex $u \in V$ with three vertices u, u', and u''. Think of u as the *entry vertex* and u'' as the *exit vertex*. We create undirected edges (u, u') and (u', u''). Also, for each directed edge $(u, v) \in E$, we create the undirected edge (v'', u), from v's exit vertex to u's entry vertex.. Let G' be the resulting graph (see Fig. 3(b)).

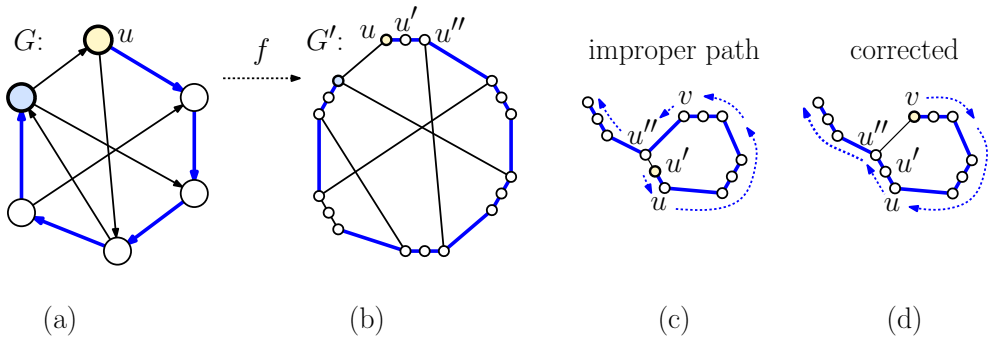

Figure 3: DHP to HP reduction.

It is not hard to see that if G has a Hamiltonian path, then G' will have one as well, by tracing each triple in the proper order (u, u', u''). However, the converse is not at all obvious. Could the path in G' could start at a middle vertex u', for example, which does not correspond to a path in G. We'll show that if this happens, we can reorient the path so it will start at an entry vertex and end at an exit vertex and visit all vertices along the way.

Claim: The digraph G has a Hamiltonian path if and only if the undirected graph G' has a Hamiltonian path.

Proof: ($\Rightarrow$) If G has a directed Hamiltonian path $\langle u_1, \ldots, u_n\rangle$, we assert that G' has the Hamiltonian path by replacing each vertex u_i with the triple u_i, u_i', u_i'' (see Fig. 3(b)). Since this is Hamiltonian path in G, the edge (u_{i-1}, u_i) is in G, which implies that the edge (u_{i-1}'', u_i) is in G'. By construction the edges (u_i, u_i') and (u_i', u_i'') are in G'. Therefore, this a Hamiltonian path in G'.

($\Leftarrow$) Suppose that G' has a Hamiltonian path. We first claim that we may assume that such a path starts at some vertex u and ends at some vertex v''. If not, we say that the path is *improper*. One way that a path may be improper is that it starts at a middle

vertex u'. We will assume that the next vertex on the path is the entry vertex u (see Fig. 3(c)), since a symmetrical argument applies if the path goes next to u''. Since it is Hamiltonian, the path must eventually return to the exit vertex u''. By the nature of G', the vertex v preceding u'' must be an entry vertex. We reverse the path so it starts at v, then goes to u then to u' and u''. After that it follows the original path (see Fig. 3(d)). If the path ends at a middle vertex, a similar correction can be performed.

After this correction, the path starts either at an entry or exit vertex. If it starts at an exit vertex, reverse the entire path, so it starts at an entry vertex. Henceforth, the path must follow the proper structure, entering each vertex triple at the entry vertex and leaving at an exit vertex.

Now that the path is proper, it is easy to see that it corresponds to a valid Hamiltonian path in G, since each edge between two vertices leaves on an exit vertex u'' and enters on an entry vertex v, but this implies that the directed edge (u, v) is part of the original graph.

By the way, you could make the correctness proof simpler by modifying the transformation. We can add a source vertex s to G, which is joined by directed edges from s into to all the vertices of G and a sink vertex t, which is joined by directed edges from all the vertices of G into t. This does not change whether G has a Hamiltonian path, but it simplifies the proof because it is easy to see that any Hamiltonian path in G' must have one endpoint at the entry vertex s and the other at the exit vertex t'', since these vertices have degree 1. The only way the path could be improper is if it starts at t'' and ends at s, in which case we simply reverse it.

(b) Given an undirected graph $G = (V, E)$, for each vertex $u \in V$, the algorithm creates a new vertex u' and adds the undirected edge (u, u') (see Fig. 4(a)). Let G' denote the resulting graph. The following lemma shows that this is correct.

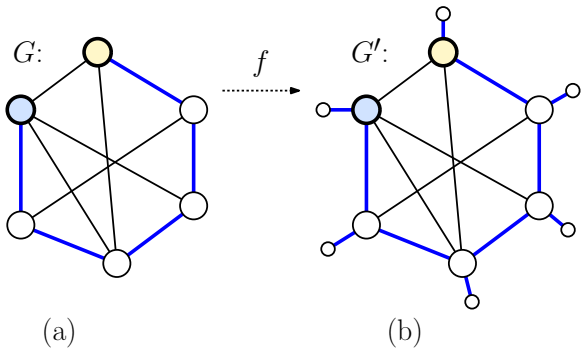

Figure 4: HP to D3ST reduction.

Claim: G has a Hamiltonian path if and only if G' has a degree-3 spanning tree.

Proof: ($\Rightarrow$) The Hamiltonian path forms a spanning tree of degree at most two in G. Adding the additional edge (u, u') to each vertex creates a spanning tree of degree at most three.

($\Leftarrow$) Observe that each of the newly added edges (u, u') must be in any spanning tree, because this is the only edge incident to u'. Removing these edges decreases the degree

of all the remaining vertices by one. These vertices are all from the original graph. Thus, we are left with a spanning tree of degree at most two. Such a spanning tree is a Hamiltonian path.

Solution 3:

(a) We claim that the optimum tour just walks around the bounding rectangle for the points. To see that this is optimal, observe that there are $2n$ points and hence $2n$ edges, and each pair of points is separated by unit distance, so any TSP tour has total length at least $2n$. This is exactly the same as the perimiter of the bounding rectangle, so this is optimal.

(b) Let's start with a_1 and travel in clockwise order around the MST. Before short-cutting, the twice-around tour has $\langle a_1, b_1, a_1, a_2, b_2, a_2, b_3, \ldots, a_n, b_n, a_n, a_1\rangle$ (see Fig. 5(a)). After short-cutting, we have $\langle a_1, b_1, a_2, b_2, b_3, \ldots, a_n, b_n, a_1\rangle$. There are $n-1$ segments, each of the form $\langle a_i, b_i, a_{i+1}\rangle$, which has an L_1 length of $1+2=3$. At the end of the tour, we have the sequence $\langle a_n, b_n, a_1\rangle$, which has an L_1 length of $1+2+(n-1)$. Thus, the overall L_1 length of this tour is $\mathrm{TA}(P(n)) = 3(n-1) + (1+2+(n-1)) = 4n-1$. Therefore, the performance ratio is

$$\lim_{n\to\infty} \frac{\mathrm{TA}(P(n))}{\mathrm{TSP}(P(n))} \;=\; \lim_{n\to\infty} \frac{4n-1}{2n} \;=\; 2.$$

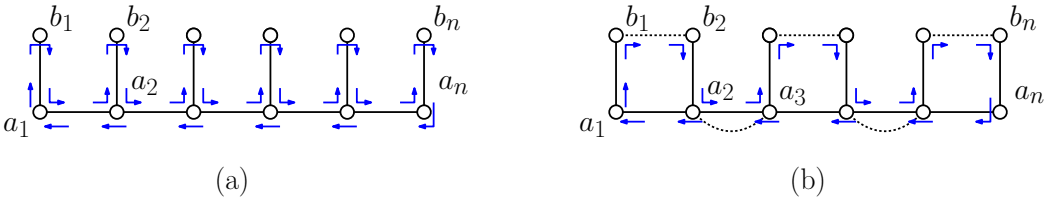

Figure 5: Approximation ratios for the metric TSP problem.

(c) The odd-degree vertices consist of the teeth of the comb (all of degree 1) and the vertices $a_2, \ldots, a_{n-1}$ (all of degree 3). Assuming that n is even, the minimum-weight perfect matching among these points connects (b_{2k-1}, b_{2k}) for $1 \le k \le n/2$ and (a_{2k}, a_{2k+1}), for $1 \le k \le n/2-1$. In total, there are $n-1$ edges of length 1, for a total weight of $n-1$.

The Eulerian circuit has total weight equal to the weight of the MST, which is $2n-1$ plus the $n-1$ from the matching, which is $3n-2$. The resulting Eulerian circuit consists of two pieces. First, a path that zig-zags from left to right: $\langle a_1, b_1, b_2, a_2, a_3, b_3, b_4, a_4, \ldots, b_n, a_n\rangle$. The second is a straight-line path from a_n back to a_1, going through all the a-points. Short-cutting does not affect the length of either of these paths, so the L_1 length of the Eulerian circuit after short-cutting is still $3n-2$. Therefore, the performance ratio is

$$\lim_{n\to\infty} \frac{\mathrm{TA}(P(n))}{\mathrm{TSP}(P(n))} \;=\; \lim_{n\to\infty} \frac{3n-2}{2n} \;=\; \frac{3}{2}.$$

Solution 4: This problem is known as the *recitilinear minimum Steiner tree problem* (RMST).

(a) Consider the following point set $P = \{(0,1),(1,0),(1,2)\}$ (see Fig. 6(a)). $\text{MST}(P)$ consists of two edges, each of length 2, for a total weight of 4 (see Fig. 6(b)). In contrast, the minimum connector places a vertex at $(1,1)$, and has three line segments going to each of the three points, for a total weight of 3 (see Fig. 6(c)).

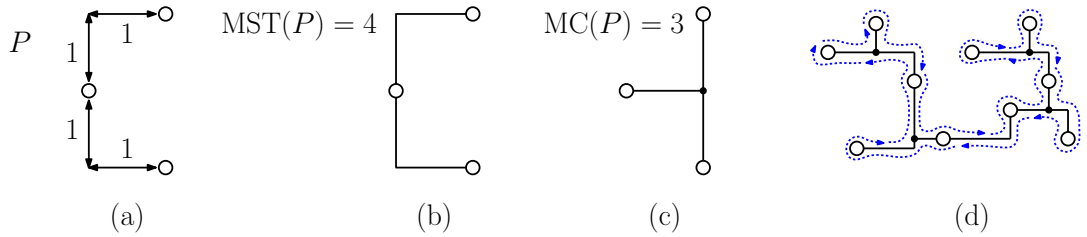

Figure 6: Minimum connector.

(b) In class, we showed that in any metric space $\text{MST}(P) \leq \text{TSP}(P)$ (since the TSP minus one edge is a spanning tree). We will use a similar argument to show that $\text{TSP}(P) \leq 2 \cdot \text{MC}(P)$. First, observe that the minimum-connector must be a tree, since we could eliminate any cycle while maintaining connectivity and decreasing the total weight. Since it is connected, we can apply the twice-around tour and then apply short-cutting to obtain a TSP tour whose cost is at most twice that of $\text{MC}(P)$. Therefore, we have

$$\text{MST}(P) \;\leq\; \text{TSP}(P) \;\leq\; 2 \cdot \text{MC}(P),$$

implying that

$$\frac{\text{MST}(P)}{\text{MC}(P)} \;\leq\; 2.$$

Solution 5: Let's first derive an exact algorithm. Recall that in the exact algorithm presented in class, after phase i, for $0 \leq i \leq n$, the list L contains all the possible sums that can be made from the elements $\{x_1, \ldots, x_i\}$. Rather than maintain a single list L, we will maintain $m+1$ lists L_j, for $0 \leq j \leq m$. After the ith phase of the algorithm L_j stores all the possible sums of the elements $\{x_1, \ldots, x_i\}$, but with the additional condition that there are exactly j hazardous elements in the sum. We modify the algorithm as follows.

- If x_i is non-hazardous, then we update each list L_j following the standard process. That is, $L_j \leftarrow L_j \cup (L_j + x_i)$, for $0 \leq j \leq m$.
- If it is hazardous, then when adding x_i, we need to promote from L_{j-1} to L_j. That is, $L_j \leftarrow L_j \cup (L_{j-1} + x_i)$, for $1 \leq j \leq m$.

The final result is the max among all the lists. The correctness follows as with the standard algorithm. We can convert this into an approximation algorithm by applying the compression process. The running time is larger by a factor of m which is at most n. Since the original algorithm ran in $O((n^2 \log t)/\varepsilon))$, this runs in time $O((n^3 \log t)/\varepsilon))$. The algorithm is presented in the following code block.

Solution to the Challenge Problem:

```
approx-hss(x[1..n], t, eps) {              // approx subset sum with hazards
    delta = eps/n                          // approx factor per stage
    for (j = 0 to m) L[j] = <0>            // basis case - no elements in sum
    for (i = 1 to n) {                     // consider item x[i]
        if (x[i] is not hazardous) {
            for (j = 0 to m)
                L[j] = merge(L[j], L[j] + x[i])     // standard update rule
        } else {                                    // x[i] is hazardous
            for (j = 1 to m)
                L[j] = merge(L[j], L[j-1] + x[i])   // promote when adding x[i]
        }
        L = compress(L, delta, t)          // ...compress similar values and items > t
    }
    return the largest element in L[0], ..., L[m]
}
```

(a) We will show that the triangle inequality holds. The other two are trivial. Let $p = (x, y)$ and $p = (x', y')$, since $\|p' - p\|_1$ is the some of the L_1 distances of the x- and y-components, it suffices to prove the triangle inequality for each component individually. Given three real numbers a, b, c, we want to show that

$$|c - a| \leq |c - b| + |b - a|.$$

We may assume without loss of generality that $a \leq c$. There are two cases, depending on whether b lies within the interval $[a, c]$ or outside. If it within the interval, then $|c-a| = |c-b|+|b-a|$, and we are done. If not, let's assume that $b < a$. Then $|c-a| \leq |c-b| \leq |c-b|+|b-a$, as desired. The case where $b > c$ is similar.

(b) We assert that we never need to use any edges other than edges of length 1. The reason is that the weight of the minimum spanning tree is at most $2n - 1$ (as witnessed by the comb) and every pair of points is at least 1 unit apart, meaning we need at least this much weight in any spanning tree. Since there are $2n - 1$ edges in the tree, all the edges are of weight 1.

We will maintain two quantities. (We'll see later why this is needed.) First, let $T(n)$ denote the number of distinct minimum spanning trees on $P(n)$, and second, let $S(n)$ denote the number of spanning forests on $P(n)$ that consists of two connected components, where a_n and b_n are in different connected components. Let's just call this a 2SF for short.

To derive $T(n)$, observe that there is a unique basis case when $n = 1$ (the single edge (a_1, b_1)). Otherwise, if $n \geq 2$, we can create MST(n) in two different ways. First, we can take any minimum spanning tree MST$(n - 1)$ and add one of the three two-edge structures shown in Fig. 7(a), (b), and (c). But this is not the only way to build a spanning tree. We can also take any 2SF$(n - 1)$ structure, and add the three-edge structure shown in Fig. 7(d). (This is why we need $S(n)$.)

Thus, we have the following recursion:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n-1) + S(n-1) & \text{otherwise.} \end{cases}$$

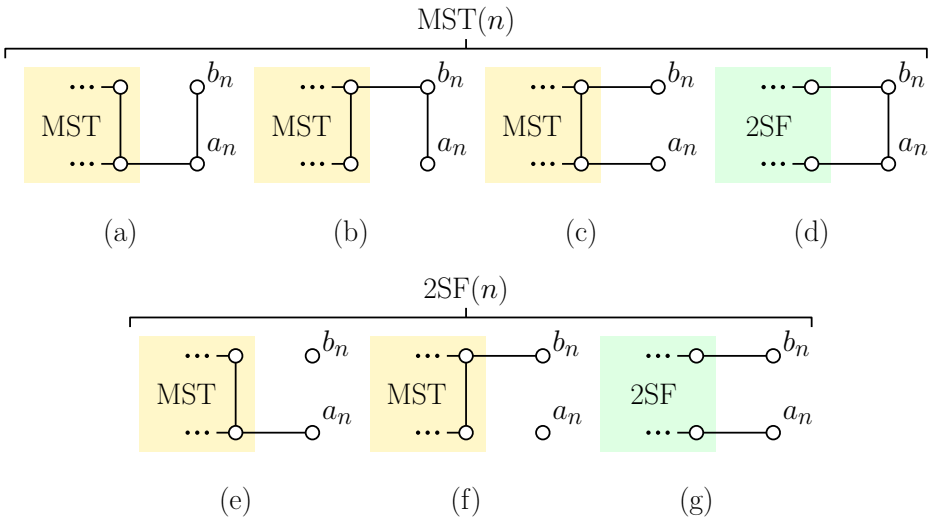

Figure 7: Minimum spanning trees.

Next, to derive $S(n)$, observe that we also have a unique basis case (two isolated vertices). Otherwise, we can create 2SF(n) in two different ways. We can either add a single horizontal edge from MST($n-1$) to either a_n or b_n (see Fig. 7(e), (f)), or we can add two horizontal edges to 2SF($n-1$) (see Fig. 7(g)). Thus, we have

$$S(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + S(n-1) & \text{otherwise.} \end{cases}$$

To get some more insight, I worked out the first few terms $T(n) = \langle 1, 4, 15, 56, 209, 780, 2911 \rangle$, and looked it up on the Online Encyclopedia of Integer Sequences, it revealed that both recurrences satisfy the same rule, $T(n) = 4T(n-1) - T(n-2)$ and $S(n) = 4S(n-1) - S(n-2)$, albeit with two different basis cases for $n = 0$, namely, $T(0) = 1$ and $S(0) = 1$. This can be solved in closed form through the use of generating functions. (Which is beyond the scope of this class.) The function grows roughly as $T(n) = (2+\sqrt{3})^n \approx 3.73^n$.

Following these formulas, we have $T(1) = S(1) = 1$. $T(2) = 3 + 1 = 4$, $S(2) = 2 + 1 = 3$, and $T(3) = 12 + 3 = 15$. These 15 trees on $P(3)$ are shown in Fig. 8.

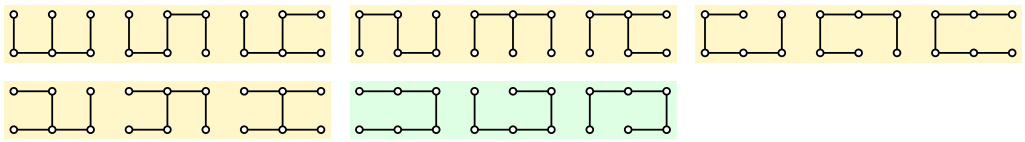

Figure 8: Minimum spanning trees.