CS526 hw2 P1

(a) Pick a root r in T. Note $X_r \neq \emptyset$ (else we would have $X_r \subseteq X_c$ for any child c of r). So we can pick a vertex $v_r \in X_r$. Now for any node $i \neq r$, it has a parent p. Since $X_i \not\subseteq X_p$, we can pick a vertex $v_i \in X_i - X_p$.

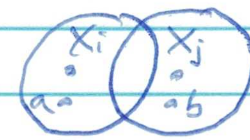
By property (3) of tree decomps, we can see all these v_i's (including v_r) are distinct. Therefore #(tree nodes) $\leq$ #(graph vertices) $= |V|$.

(b) Suppose $x \in A$, $y \in B$, and $\{x,y\}$ is an edge of G. As argued in discussion, it is enough to show x or y is in $S = X_i \cap X_j$. Furthermore, we also saw $S = A \cap B$.

By property (2) of tree decomps, some bag X_ℓ contains $\{x,y\}$. Note ℓ is in either T_A or T_B, suppose it is in T_A (T_B argument is similar). Then $y \in X_\ell \subseteq A$, so $y \in A \cap B = S$.

(c) While there are adjacent bags $X_i - X_j$ with $|X_i| < |X_j|$, copy a new vertex from X_j to X_i. This eventually ends with all bags the same size $(k+1)$, and it does not violate property (3).

Next, while there are adjacent bags $X_i - X_j$ with $|X_i - X_j| = |X_j - X_i| \geq 2$, do the following:



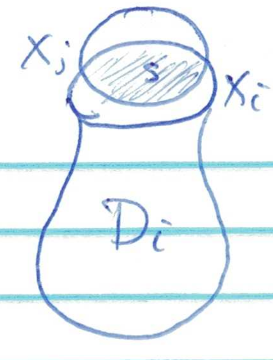
① pick $a \in X_i - X_j$ and $b \in X_j - X_i$

② create new tree node ℓ with $X_\ell = X_i - \{a\} + \{b\}$

③ insert ℓ between i and j in T: $X_i - X_\ell - X_j$

This eventually terminates with all $|X_i - X_j| = 1$. And again, we do not violate the tree decomp properties.

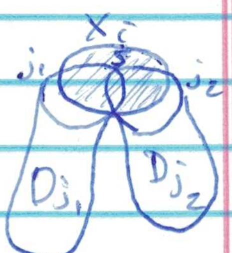
hw2 P2



(a) $B(s,i,j) = \bigvee_{s'} A(s',i)$

where s' ranges over all "extensions" of s to all of X_i. In other words, $s'(v) = s(v)$ for all $v \in X_i \cap X_j$.

(b) $A(s,i) = \text{check}(s) \wedge \bigwedge_{\substack{\text{child } j \\ \text{of } i}} B(s|_{X_i \cap X_j}, j, i)$



Note j ranges over children of i, and this function is the restriction of s to the domain $X_i \cap X_j$.
If i has no children, this is just $A(s,i) = \text{check}(s)$.

(c) Assuming the tree decomposition is smooth, the B Formula takes time $O(1)$ [just 3 cases], and the A Formula takes time $O(k^2 + k \cdot \deg(i))$, where $\deg(i)$ is the degree of node i in T.
Also there are $O(3^k)$ choices for each s.

Summing over all subproblems, and using the Fact that $\sum_i \deg(i) = O(n)$, we get
$O(3^k \cdot k^2 \cdot n)$ time. Or maybe
$O(3^k \cdot k^3 \cdot n)$ since our "keys" have size $O(k)$.

Remark: Any $O(3^k \cdot \text{poly}(k) \cdot n)$ is Fine here!

CS526 hw2 P3 (DPV 6.13)

I'll use $\{1, 2, \ldots, n\}$ to name cards, with the values $v_1, v_2, \ldots, v_n$. Call players I and II (I plays first and whenever an even # remain.)

(a) $n=4$, $\vec{v} = (1,1,3,2)$. Greedy I would take $v_4 = 2$ and then get 1 more, but it is better to take $v_1 = 1$ and then $v_3 = 3$. $(2+1 < 1+3)$

(b) For $1 \le i \le j \le n$, note $(v_i, \ldots, v_j)$ describes a "remaining game" (cards still on the table). Define $V(i,j)$ = "max sum of cards I can earn from this point, assuming both I and II play optimally". Note I wants to maximize V, II wants to minimize.

We compute all the $V(i,j)$ in table $V[i,j]$:

```
ComputeV (v1, v2, ..., vn):
    for i = 1 to n
        V[i,i] = 0   // since last card goes to II
    for l = 2 to n   // # of remaining cards
      for i = 1 to n-l+1
        j = i+l-1
        if l is even   // I picks vi or vj
          V[i,j] = max(vi + V[i+1,j], vj + V[i,j-1])
        else   // II picks vi or vj
          V[i,j] = min(V[i+1,j], V[i,j-1])
```

<u>First move:</u> IF $v_1 + V[2,n] > v_n + V[1,n-1]$, then I picks v_1, else I picks v_n.

CS526 hw2 P4 (DPV 5.22 or 5.23)

Reminder: We have graph $G=(V,E)$, edge weights w, MST $T=(V,E')$, and now we want to change weight of one e to $\hat{w}(e)$.
How can we find the new MST T' in each case?

(a) $e \notin E'$ and $\hat{w}(e) > w(e)$: no change! $T' = T$

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$:

Add e to T. Find cycle C in $T+e$.
Find the heaviest edge f on C.
Let $T' = T + e - f$.
Remark: This takes $O(|V|)$ time.

(c) $e \in E'$ and $\hat{w}(e) < w(e)$: no change! $T' = T$

(d) $e \in E'$ and $\hat{w}(e) > w(e)$:

Identify the two components of $T - e$ (color vertices with 0 or 1).
Looking at all edges of G, finding the lightest edge f between the two components.
Let $T' = T - e + f$.
Remark: This takes $O(|V|+|E|)$ time.

Note: You did not have to argue correctness.
Also $T' = T$ is possible in both (b) and (d).

CS526 hw2 P5

Solution(G, k):

① Using $w(\text{red})=0$, $w(\text{blue})=1$, compute MST T_2 with maximum # of red edges, k_2.
If $k_2 < k$, return "NO SOLUTION".

② Using $w(\text{red})=1$, $w(\text{blue})=0$, compute MST T_1 with minimum # of red edges, k_1.
If $k_1 > k$, return "NO SOLUTION".

③ $T = T_1$
while T has $< k$ red edges:

$O(V)$ {
Pick an edge $e \in T_2 - T$
Find cycle C in $T+e$, and pick an edge $f \in C$ such that $f \notin T_2$.
(f must exist, because T_2 has no cycle.)
$T \leftarrow T + e - f$
}

Return T.

Loop ③ must terminate because $|T \Delta T_2|$ (the "hamming distance" between T and T_2 as sets of edges) decreases on each step. Also #red (T) increases by at most 1 on each step, so we exit loop with #red$(T)=k$.

Note ① and ② can be done in $O(V^2)$ time (by Prim's algo). Also each iteration of ③ takes $O(V)$ time (cycle finding, edge table), so it is $O(V^2)$ overall.

Remark: this adapts to any matroid.