Cell tower: Let c be first position Greedy and optimal differ. Let t be house at a_c-1. Alg always places tower 1 mi out from uncovered house + optimal and greedy were same before c, so no tower before a_c that covers t. Then o_c and a_c must cover t. Since a_c=t+1 and $a_c \neq o_c$, $o_c < a_c$ to cover t. o_c covers all houses a_c covers then. Can exchange until O = A.

Coin change: best (b_{50}, b_{25}, b_{10}, b_5, b_1), Greedy (a_{50},...,a_1). Show $\sum a_i$ smaller. Since best is not greedy, some point threw will be fewer coins of some denom. **(1)** if $b_{50}<a_{50}$ then rest must make up missing 50: $25b_{25}+10b_{10}+5b_5+b_1 \geq 50$: If $b_{25} \geq 2$ replace with half dollar, b=1 forces more dimes and nickels->replace with half dollar… **(2)** $b_{50}=a_{50}$ and $b_{25}<a_{25}$ then $10b_{10}+5b_5+b_1 \geq 25$: if $b_{10} \geq 3$ replace with 1 quarter 1 nickel… repeat until $b_5=c_5$ must all be the same

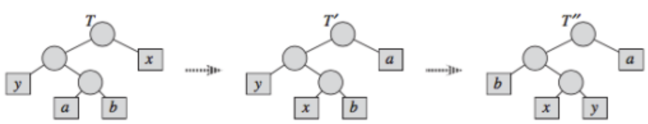
Union: Sort by start index, start merging.

Huffman

for i = 1 to n − 1 do

1. z ← allocateNode() 2. x ← lef t[z] ← DeleteMin(Q) 3. y ← right[z] ← DeleteMin(Q)
4. f[z] ← f[x] + f[y] 5. Insert(Q, z)

Proof: Lemma: Suppose x and y are two letters of lowest frequency. Then, there exists an optimal prefix code in which codewords for x and y have the same (and maximum) length and they differ only in the last bit.



Proof. Start with an optimal prefix code tree T, and modify it so x and y are sibling leaves of max depth, without increasing total cost. • In modified tree, x and y have the same code length, different only in the last bit. • Assume optimal tree does not satisfy the claim, and suppose that a and b are the two characters that are sibling leaves of max depth in T. • Without loss of generality, assume that f(a) ≤ f(b) and f(x) ≤ f(y) • We have f(x) ≤ f(a) and f(y) ≤ f(b). (x, y, a, b need not all be distinct.)

Horn:

- First transform T into T' by swapping the positions of x and a.
- Since $d_T(a) \geq d_T(x)$ and $f(a) \geq f(x)$, swap does not increase $freq \times depth$ cost:

$$\begin{aligned} B(T) - B(T') &= \sum_p [f(p) d_T(p)] - \sum_p [f(p) d'_T(p)] \\ &= [f(x)d_T(x) + f(a)d_T(a)] - [f(x)d'_T(x) + f(a)d'_T(a)] \\ &= [f(x)d_T(x) + f(a)d_T(a)] - [f(x)d_T(a) + f(a)d_T(x)] \\ &= [f(a) - f(x)] \times [d_T(a) - d_T(x)] \\ &\geq 0 \end{aligned}$$

- Next, transform T' into T'' by exchanging y and b, which also does not increase cost.
- So, we get that $B(T'') \leq B(T') \leq B(T)$. If T was optimal, so is T'', but in T'' x and y are sibling leaves at the max depth.
- This completes the proof of the lemma.

- **Proof of optimality.**
- Let T_1 be the optimal tree (induction) for $C + \{z\} - \{x, y\}$.
- We obtain our final tree T by attaching leaves x, y as children of z.
- What is the connection between costs of $B(T)$ and $B(T_1)$?
- For all $p \neq x, y$, depth is the same in both trees, so no difference. For x, y, we have $d_T(x) = d_T(y) = d_{T_1}(z) + 1$. So, the cost increase from modifying T_1 to T is:

$$B(T) - B(T_1) = f(x) + f(y)$$

because

$$f(x)d_T(x) + f(y)d_T(y) = [f(x) + f(y)] \times [d_{T_1}(z) + 1] = f(z)d_{T_1}(z) + [f(x) + f(y)]$$

- The rest of the argument is via contradiction. Suppose T is not an optimal prefix code, and another tree T' is claimed to be optimal, meaning $B(T') < B(T)$.
- By previous lemma, T' has x and y as siblings. Imagine replacing parent of x, y with a new leaf z, with freq. $f(z) = f(x) + f(y)$, and call this new tree T'_1.
- Then,

$$B(T'_1) = B(T') - f(x) - f(y) < B(T) - f(x) - f(y) < B(T_1)$$

which contradicts the claim that T_1 is an optimal prefix code for $C' = C + \{z\} - \{x, y\}$. End of proof.

FastGreedyHorn(φ): 1. Set v to False for each variable v in φ. 2. Set W := {v : v appears on the right-hand side of an empty implication}. 3. While W

6= ∅, do: 4. Take (and delete) v from W. 5. Set v to True. 6. For each clause c where v appears on the left-hand side, do: 7. Delete v from the left-hand side of c. 8. If this makes c into an empty implication, add the variable on the right-hand side of c into W (if it is not already in W). 9. Return the current truth assignment.

Dijkstra:1. Argue that at any time d(v) is the shortest path distance to v, for all v ∈ S. 2. Consider the instant when node v is chosen by the algorithm. Let (u, v) be the edge, with u ∈ S, that is incident to v. 3. Suppose, for the sake of contradiction, that d(u) + cost(u, v) is not the shortest path distance to v. Instead a shorter path P exists to v. 4. Since that path starts at s, it has to leave S at some node. Let x be that node, and let y 6∈ S be the edge that goes from S to S. 5. So our claim is that length(P) = d(x) +cost(x, y) +length(y, v) is shorter than d(u) + cost(u, v). But note that the algorithm chose v over y, so it must be that d(u) + cost(u, v) ≤ d(x) + cost(x, y). 6. In addition, since length(y, v) > 0, this contradicts our hypothesis that P is shorter than d(u) + cost(u, v). 7. Thus, the d(v) = d(u) + cost(u, v) is correct shortest path distance.

1. Let S be the set of *explored nodes*.
2. Let $d(u)$ be the shortest path distance from s to u, for each $u \in S$.
3. Initially $S = \{s\}$, $d(s) = 0$, and $d(u) = \infty$, for all $u \neq s$.
4. While $S \neq V$ do
   (a) Select $v \notin S$ with the minimum value of
   $$d'(v) = \min_{(u,v), u \in S} \{d(u) + cost(u, v)\}$$
   (b) Add v to S, set $d(v) = d'(v)$.

Kruskal: (v,w) first edge that differs, Let S be all reachable from v, then w not in S or else wouldnt consider. OPT has path from v to w but not through (v,w). Since v,w disconnected in K, OPT has some edge that crosses from S to !S called (x,y). (x,y) not added to K yet since y is not reachable from v (not in S). C(x,y) > C(v,w), can swap, more optimal, contra.

Prim: Let T be the spanning tree found by Prim's algorithm and T* be the MST of G. We will prove T = T* by contradiction. Assume T ≠ T*. Therefore, T – T* ≠ Ø. Let (u, v) be any edge in T – T*. When (u, v) was added to T, it was the least-cost edge crossing some cut (S, V – S). Since T* is an MST, there must be a path from u to v in T*. This path begins in S and ends in V – S, so there must be some edge (x, y) along that path where x ∈ S and y ∈ V – S. Since (u, v) is the leastcost edge crossing (S, V – S), we have c(u, v) < c(x, y). Let T*' = T* ∪ {(u, v)} – {(x, y)}. Since (x, y) is on the cycle formed by adding (u, v), this means T*' is a spanning tree. However, c(T*') = c(T*) + c(u, v) – c(x, y) < c(T*), contradicting that T* is an MST. We have reached a contradiction, so our assumption must have been wrong. Thus T = T*, so T is an MST.