Homework 4

Thomas Kim tsk389 51835

David Munoz dam2989 51840

CS331 Algorithms and Complexity

Problem Q1. Given n samples and unambiguous ImgComp, design a divide and conquer algorithm which can report whether more than 50% of samples belong to a single class. Prove correctness and runtime.

main

$S' =$ signal-to-noise(S)
if $\frac{|S'|}{|S|} > .5$ **then**
  return $"YES"$
else
  return $"NO"$
end if

signal-to-noise(S)

Let $S := \{s_1, s_2, \ldots, s_n\}$ be the set of images used as input to ImgComp
Let $A := \{s_1, s_2, \ldots, s_{\frac{n}{2}}\}, B = S - A$
if $|A| = |B| = 1 \wedge ImgComp(s_1, s_2) = "YES"$ **then**
  return $\{s_1, s_2\}$
else if $|A| = |B| = 1 \wedge ImgComp(s_1, s_2) = "NO"$ **then**
  return $\{s_1\}$
else
  $A' :=$ signal-to-noise(A)
  $B' :=$ signal-to-noise(B)
  $A'' := A'$
  $B'' := B'$
  for $\forall b_i \in B$ **do**
    if $ImgComp(a_1, b_i) = "YES"$ **then**
      $B'' := B'' \cup \{b_i\}$
    end if
  end for
  for $\forall a_i \in A$ **do**
    if $ImgComp(b_1, a_i) = "YES"$ **then**
      $A'' := A'' \cup \{a_i\}$
    end if
  end for
  return $max(A'', B'')$
end if

Proof of correctness

Proof. Claim: If more than 50% of S belongs to one set C, $a \in A' \wedge a \in C \vee b \in B' \wedge b \in C$ and signal-to-noise(S) will return C
Base case: $|S| = 4, |A| = |B| = 2$
Suppose more than 50% of S belongs to some category C. This means $|C| > \frac{S}{2} \implies |C| > 2$.
By the pigeonhole principle (either A or B will get a matching pair because at least 3 out of 4 match and there are 2 subsets), $|A'| = 2 \vee |B'| = 2$.
Without loss of generality, suppose $|A'| = 2$. If A' does not contain some $a \in C$, then $|C| \leq |S - A'| = 2 \implies$ contradiction.
Given the correctness of A' and/or B', lines

Induction:
Suppose $a \in A' \wedge b \in B' \wedge A' \cap C = \{\} \wedge B' \cap C = \{\}$

$\square$

Proof of Runtime

Proof. signal-to-noise(S) creates 2 subproblems of size $\frac{n}{2}$ by line
signal-to-noise(S) calls $ImgComp$ n times by lines , since $|A'| \leq |A| = \frac{n}{2} \wedge |B'| \leq |B| = \frac{n}{2}$
Therefore, the recurrence for signal-to-noise(S) is $T(n) = 2T(\frac{n}{2}) + \theta(n)$ By master theorem, $f(n) = \theta(n) = \theta(n^1 log^0(n)) \implies T(n) = \theta(nlog(n))$ $\square$

Problem Q2. $FindLocalMinimum(G)$

Define $quad(v)$ to be (the set of vertices in the same quadrant of G as v) unioned with (any elements of B which border the previous set).
Let B, C, T, TC be defined on G as specified in the problem.
$minVert \leftarrow min_{v \in S \cup T} F(v)$
if $minVert \in S \vee minVert \in TC$ **then**
    return $minVert$
else
    return $FindLocalMinimum(quad(minVert))$
end if

Lemma 1: For all such grids G, there exists a local minimum.

Proof. There are a finite number of squares each with unique weight w_i.
Let the set $W = \{w_1, w_2, \ldots, w_i\}$
By the well-ordering principle, there exists a least element of W.
By definition, this least element is less than each of its neighbors, because it is less than every other weight in the set.
Thus, it is a local (global) minimum.

$\square$

Base case: Note that there are no bases cases by lemma 1.
Proof of correctness:

Proof. Suppose there is no local minimum when $S' \cup T' = G'$, where G' is the subgrid in the final iteration.
Take v_c to be the center vertex of G'.
Note that in the previous iteration, v_c must have been in T, since it couldn't have been in $S \cup TC$ and $S' \cup T' = G'$ This means v_c is the smallest element in G'
This means v_c is a local minimum.

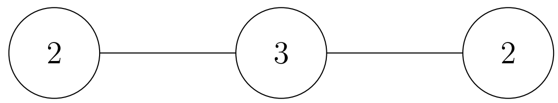
$\square$

Proof of runtime:

Proof. The recurrence associated with this algorithm is $T(n) = T(\frac{n}{2}) + \theta(n)$
Expanding this recurrence, $T(n) = T(\frac{n}{4}) + c\frac{n}{2} + n$
$= \sum_{i=1}^{n} c\frac{n}{2^i}$
Therefore the total runtime is $\theta(n)$

$\square$

Problem Q3(a).



In this example, the middle node will be chosen and the two side nodes discarded for a total weight of 3.
If the middle node was discarded and the two side nodes chosen, the total weight would have been $4 > 3$.

Problem Q3(b).

In this example, the maximum weight independent set includes nodes v_1 and v_4, but 1 is odd and 4 is even.

Problem Q3(c). Let the set of independent vertices $S_i \subset \{v_1, v_2, v_3, \ldots, v_i\}, i \leq n$ have some weight W_i
Let each vertex v_i have some weight w_i
Suppose S_i includes vertex v_i. S_i can therefore cannot include v_{i-1}, so $W_i = W_{i-2} + w_i$
Suppose S_i does not include v_i. $W_i = W_{i-1}$
Base case(s): $W_0 = 0, W_1 = w_1$
Algorithm: $W_i = max(W_{i-1}, W_{i-2} + w_i)$
return W_n
Memoization: W_j only has to be computed once for each unique j.
Proof of correctness:

Proof. Suppose W_n is not the maximum weight.
This means either W_{i-1} or W_{i-2} wasn't maximized.
This continues, with some W_{i-k} not being maximized.
However, W_0 and W_1 are always maximized by definition.
Thus, $W_{i-k}, k = i$ must be maximized.
This contradicts the original assumption, since each step is guaranteed to maximize the weight if the previous step is maximized.

□

Proof of runtime:

Proof. $W_0, W_1, \ldots, W_n$ will all be calculated exactly once due to memoization.
The time required to calculate W_i is $O(1)$, since it is a single comparison and a single addition.
Therefore, the total complexity is $O(n)$.

□