Recurrence Relations

Let $T(n)$ be the number of strings in $\{0,1,2\}^n$ that do NOT contain two consecutive zeros. Write a recurrence relation for $T(n)$.

Recurrence with an Algorithm

Algorithm 1: `MergeSort`(C, n)

```
Input  : Array of C of length n (where n is a power of 2)
Output: Sorted array containing all elements of C
if n==1 then
    return C;
end
A=MergeSort(C[1 : n/2], n/2);
B=MergeSort(C[n/2 + 1 : n/2], n/2);
p_A = 1;
p_B = 1;
Increase length of A and B by 1 each, and set final element of each array to ∞;
for k = 1 to n do
    if A[p_A] < B[p_B] then
        C[k] = A[p_A];
        p_A+ = 1;
    else
        C[k] = B[p_B];
        p_B+ = 1;
    end
end
```

- Create a recurrence relation for the runtime of MergeSort, and evaluate using the iterative method.

Indicator Random Variables

Consider an ordered list of the elements $\{1,2,3, \dots, n\}$. An inversion is a pair (i, j) where $i < j$ but j precedes i in the list. For example if we consider the ordered list $(3,1,4,2)$ of the elements $\{1,2,3,4\}$, there are 3 inversions: $(1,3)$, $(2,3)$, $(2,4)$. But for example $(1,2)$ is not inverted. If an ordering is chosen with equal probability from among all possible orderings, what is the average number of inversions?

1. What is the sample space and key random variable?
2. Break key random variable into sum of indicator random variables
3. Use linearity of expectation.
4. Use property of indicator random variables to evaluate terms in the sum, and add everything up.

Graph Pseudocode

Write pseudocode to determine if a graph has an edge (u,v) but not (v,u). In other words, is the graph directed? You can choose either Adjacency Matrix or Adjacency List…but think about which is easier before you start! What is the runtime of your algorithm?

Probability of an Event:

How should you calculate the probability of an event it

- All elements of the sample space are equally likely?
- If elements of the sample space are not equally likely?

Recurrence Relation Solution

Let $T(n)$ be the number of strings in $\{0,1,2\}^n$ that do NOT contain two consecutive zeros. Write a recurrence relation for $T(n)$.

3 options: _ _ _ _ _ 1 or _ _ _ _ _ 2 or _ _ _ _ _ 0

- If end in 1 or 2, need there to not be consecutive zeros in the first n-1 positions. There are $T(n-1)$ ways of doing this for each.
- If end in 0, we need the second to last digit to be a 1 or 2. Otherwise we have two consecutive 0's. Then for the remaining n-2 digits, we need there to be no two consecutive 0's. There are $T(n-2)$ ways of doing this. So using the product rule, there are $2T(n-2)$ ways.
- Using the sum rule: $T(n) = 2\left(\mathrm{T}(\mathrm{n}-1) + \mathrm{T}(\mathrm{n}-2)\right)$

Need two base cases to avoid falling off the ladder: $T(1) = 3, T(2) = 8$.

Recurrence with an Algorithm Solution

Algorithm 1: `MergeSort`(C, n)

Input : Array of C of length n (where n is a power of 2)
Output: Sorted array containing all elements of C

if n==*1* **then**
  return C;
end
A=MergeSort$(C[1 : n/2], n/2)$;
B=MergeSort$(C[n/2 + 1 : n/2], n/2)$;
$p_A = 1$;
$p_B = 1$;
Increase length of A and B by 1 each, and set final element of each array to ∞;
for $k = 1$ **to** n **do**
  if $A[p_A] < B[p_B]$ **then**
    $C[k] = A[p_A]$;
    $p_A+ = 1$;
  else
    $C[k] = B[p_B]$;
    $p_B+ = 1$;
  end
end

- Create a recurrence relation for the runtime of MergeSort, and evaluate using the iterative method.

- $T(n) = O(n) + 2T\left(\frac{n}{2}\right), T(1) = O(1)$
- Using iterative method, pattern is $T(n) = 2^k T\left(\frac{n}{2^k}\right) + O(nk)$. Plug in $k = \log_2 n$ to get $T(n) = n + O(n \log_2 n) = O(n \log_2 n)$

Indicator Random Variables Solution

1. What is the sample space and key random variable?
   - Sample space is set of $n!$ possible permutations of n elements. Random variable X is the number of inversions in a single permutation
2. Break key random variable into sum of indicator random variables
   - Let X_{ij} take value 1 if there is an inversion between i, j, and 0 else. Then $X = \sum_{ij} X_{ij}$
3. $E[X] = \sum_{ij} E[X_{ij}]$
4. $E[X] = \sum_{ij} \Pr[there\ is\ an\ inversion\ of\ \ i,j]$. Any two elements are equally likely to be inverted or not! So $\Pr[there\ is\ an\ inversion\ of\ \ i,j] = \frac{1}{2}$. And hence

$$E[X] = \sum_{ij} C(n,2)/2 = n(n-1)/4$$

Graph Pseudocode Solution

Write pseudocode to determine if a graph has an edge (u,v) but not (v,u). In other words, is the graph directed? You can choose either Adjacency Matrix or Adjacency List…but think about which is easier before you start! What is the runtime of your algorithm?

Use Adjacency Matrix for $G = (V, E)$, because just need to check opposite sides of the diagonal. This approach uses $O(|V|^2)$ time.

- For $u \in V$:
  - For $v \in V$:
    - If $A[u, v]$ does not equal $A[v, u]$ return True

Probability of an Event Solution

How should you calculate the probability of an event it

1. All elements of the sample space are equally likely?
2. If elements of the sample space are not equally likely?

1. Count the number of elements in the event and divide by the number of elements in the sample space.
2. Use a tree to calculate the probability of different elements of the sample space. Add up the probability of all of the elements in the event.