Expected Behavior of Algorithms

- Estimating the average complexity of some algorithms based on the distribution of data.
- The algorithms are O of some function and not Θ.
- Examples: linear search, quicksort.
- **Expected value** for a variable X for which we know the probability $f_x(x)$ for every possible outcome x is
  $E(X) = \Sigma_{\forall x} (x \cdot f_x(x))$

Linear Search

```
template <class otype>
int location_by_linear_search
  (const otype a[], const otype
  &target, int n)
{
  int i=0;
  while (i<n && a[i] != target)
      ++i;
  return i;
}
```

Linear Search

- Searching for a target in an array, and stopping when we find the value.
- $O(1) \leq T(n) \leq O(n)$
- Suppose the target is in the array and is equally likely to be in any of the positions from 0 to n-1.
- What is the expected time for a successful search?
- What is the expected time for an unsuccessful search? $\Theta(n)$.

Successful Search

- Let X be the variable representing the number of iterations for a successful search.
- For any x, $0 \leq x \leq n-1$,
  $P(X=x) = 1/n$
- The expected value for X is the sum of all possible values multiplied by their probability:
  $E(X) = \Sigma\, x \cdot P(X=x) =$
  $0 \cdot 1/n + 1 \cdot 1/n + \ldots + (n-1) \cdot 1/n = (n-1)/2$
- $E(X) = \Theta(n)$
- The search time is another statistical variable $Y = \Theta(X)$ which means $E(Y) = \Theta(n)$.

Bubble Sort

```
template <class otype>
void simple_bubble_sort (otype a[], int n)
{
  for (int k = n;  k > 1;  --k)
    for (int j = 1; j <= k; ++j)
      if (a[j] > a[j+1]])
        swap (a[j], a[j+1]);
}
```

- What is the average number of object comparisons?
  n(n-1)/2
- What is the average number of swaps?

Number of Swaps

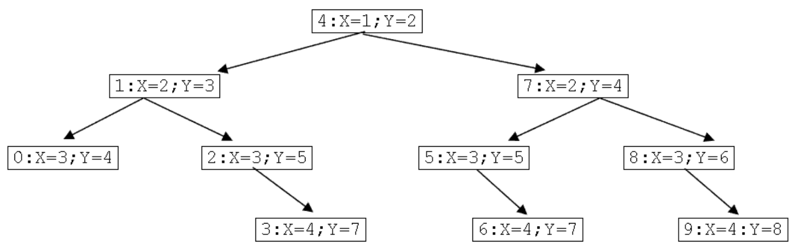
- Let Y be the total number of swaps.
- Note: every element in the array must be swapped with all those to its left that are larger than it. Let L be the random variable representing this number.
- Then $E(Y)=E(L_1)+E(L_2)+...+E(L_n)$
- L_i can take the values {0, 1, ..., i-1}. Each of them is equally probable with a probability of 1/i.
- $E(L_i) = 0/i+1/i+2/i+...+(i-1)/i = (i-1)/2$
- $E(Y)=(1-1)/2+(2-1)/2+...+(n-1)/2$
  $= n(n-1)/4$

```
void binary_search (otype a[], otype &target,
  int first, int last, bool & found, int &
  subscript)
{
   int mid;
   found = false;
   while (first <= last && !found) {
      mid = (first + last)/2;
      if (target < a[mid])
         last = mid - 1;
      else if (a[mid] < target)
         first = mid + 1;
      else
         found = true;
   }
   if (found)
      subscript = mid;
   else
      subscript = first;
}
```


Binary search

- Suppose we have a successful binary search in an array with 10 elements. What is the expected execution time?
- Let X be the number of iterations of the loop and Y the number of comparisons that will be made.
- The search starts with first=0 and last=9.

Execution Tree



Expected Time

- E(X) = 1(1/10) + 2(2/10) + 3(4/10) + 4(3/10) = 29/10 = 2.9

- E(Y) = 2(1/10) + 3(1/10) + 4(2/10) + 5(2/10) + 6(1/10) + 7(2/10) + 8(1/10) = 51/10 = 5.1

Unsuccessful Search

- $E(X) = 3(5/11) + 4(6/11) = 39/11 \cong 3.55$

- $E(Y) = 3(1/11) + 4(3/11) + 5(1/11) + 6(2/11) + 7(3/11) + 8(1/11) = 61/11 \cong 5.55$

Conditional Expected Value

- Suppose we have a successful search for a target in a sorted array by a binary search and that each position is equally likely to contain the target.
- In the first iteration, let A_1 be the event "target < A[mid]",
  A_2: "target > a[mid]" and
  A_3: "target == a[mid]"
- $P(A_3) = 1/n$
- $P(A_1) = 1/n \lfloor (n-1)/2 \rfloor$
- $P(A_2) = 1/n \lceil (n-1)/2 \rceil = 1/n \lfloor n/2 \rfloor$

Expected Value

- Let X_n be the number of object comparisons for a successful search.
- $X_n = 1 + I(A_2 \cup A_3) + Z_n$
- where $I(A_2 \cup A_3)$ is 1 if either of A_2 or A_3 occurs and Z_n is the number of comparisons done after that.
- $E(X_n) = 1+P(A_2 \cup A_3) +E(Z_n)$
  $=1+P(A_2 \cup A_3)+P(A_1)\, E(Z_n|A_1) + P(A_2)\, E(Z_n|A_2) +P(A_3)\, E(Z_n|A_3)$

Recurrence Relation

- $E(X_n) = 1 + 1/n\, (\lfloor n/2 \rfloor+1) + 1/n \lfloor (n-1)/2 \rfloor\, E(X_{\lfloor (n-1)/2 \rfloor}) + 1/n \lfloor n/2 \rfloor\, E(X_{\lfloor n/2 \rfloor}) + 0$
- If we denote $F(n)=nE(X_n)$, then this becomes
- $F(n)=n+\lfloor n/2 \rfloor+1+F(\lfloor (n-1)/2 \rfloor)+F(\lfloor n/2 \rfloor)$
- We can solve this and obtain
- $F(n) = 3/2\ n \lg n$
- => **$E(X_n) = 3/2 \lg n$**

Quicksort

- Let X_n be the number of comparisons done by the quicksort.
- $X_0=0$, $X_1=0$
- $X_n = n+1+ Y_n + Z_n$, where
- Y_n = comparisons done in the first recursive call.
- Z_n = comparisons done in the second recursive call.

$E(X_n)$

- Number of comparisons during the
- $X_n = n+1 + Y_n + Z_n$
- $E(X_n) = n+1 + E(Y_n)+E(Z_n)$
- Y_n and Z_n depend on the position of the pivot in the original array.
- Let A_k be the event that the pivot ends on the position k in the array,
  $0 \leq k \leq n-1$.
- $E(Y_n) = P(A_0)\, E(Y_n|A_0) + P(A_1)\, E(Y_n|A_1) + \ldots + P(A_{n-1})\, E(Y_n|A_{n-1})$

- $P(A_k) = 1/n$.
- $E(Y_n|A_k) = E(X_k)$
- $E(Y_n) = 1/n\ [E(X_2)+E(X_3)+...+E(X_{n-1})]$
- Same for $E(Z_n)$.
- $E(X_n) = n+1+ 2/n\ (E(X_2)+E(X_3)+...+E(X_{n-1}))$
- Denote by $F(n) = E(X_n)$
- $n\ F(n) = n^2+n+2[F(2)+F(3)+...+F(n-1)]$
- Computing $n\ F(n) - (n-1)\ F(n-1)$, we obtain
- $nF(n)-(n-1)F(n-1)=n^2-(n-1)^2+n-(n-1)+2F(n-1)$
- $nF(n) = (n + 1)\ F(n - 1) + 2n$
- Substituting $G(n) = F(n)/(n+1)$, we can solve this recurrence relation and obtain
- $F(n) = E(X_n) \cong 2\ n \ln n \cong 1.39\ n \lg n$