Week 29: Advanced Probability & Expected Value in Algorithms

Topics: - Probability Basics and Linearity of Expectation - Expected Value in Randomized Algorithms - Markov Chains and Transition Probabilities - Randomized Data Structures (Treaps, Skip Lists) - Monte Carlo and Las Vegas Algorithms - Applications: Expected Number of Steps, Random Graphs, Game Theory

Weekly Tips: - Use linearity of expectation to simplify expected value computation. - Randomized algorithms can reduce worst-case time complexity. - Markov Chains help model stochastic processes in graphs. - Monte Carlo algorithms have probabilistic correctness; Las Vegas algorithms always correct but probabilistic runtime. - Expected value often helps in counting problems and probabilistic combinatorics.

Problem 1: Expected Number of Inversions Link: CSES Expected Inversions **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    vector<double> a(n);
    for(int i=0;i<n;i++) cin>>a[i];
    double res=0;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++) res+=a[i]>a[j];
    }
    cout<<fixed<<setprecision(6)<<res<<endl;
}
```

Explanation Comments: - Compute expected inversions by counting pairs where a[i]>a[j]. - For randomized arrays, linearity of expectation applies. - Avoid explicit simulation by computing probabilities analytically.

Problem 2: Randomized QuickSort Expected Runtime Link: [Algorithm Textbook Problem Example] **Difficulty:** Advanced

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
void quicksort(vector<int>& a,int l,int r){
    if(l>=r) return;
    uniform_int_distribution<int> dist(l,r);
    int pivot=a[dist(rng)];
    int i=l,j=r;
```

```
    while(i<=j){
        while(a[i]<pivot) i++;
        while(a[j]>pivot) j--;
        if(i<=j) swap(a[i++],a[j--]);
    }
    if(l<j) quicksort(a,l,j);
    if(i<r) quicksort(a,i,r);
}
int main(){
    int n; cin>>n; vector<int> a(n);
    for(int i=0;i<n;i++) cin>>a[i];
    quicksort(a,0,n-1);
    for(int x:a) cout<<x<<' '; cout<<endl;
}
```

Explanation Comments: - Pivot chosen randomly ensures expected O(n log n) runtime. - Randomization avoids worst-case deterministic inputs. - Expected value calculation often models average runtime and probabilistic behavior.

End of Week 29 - Advanced probability and expectation techniques improve analysis of randomized algorithms. - Practice problems involving expected value, Monte Carlo, and randomized data structures for ACM-ICPC.