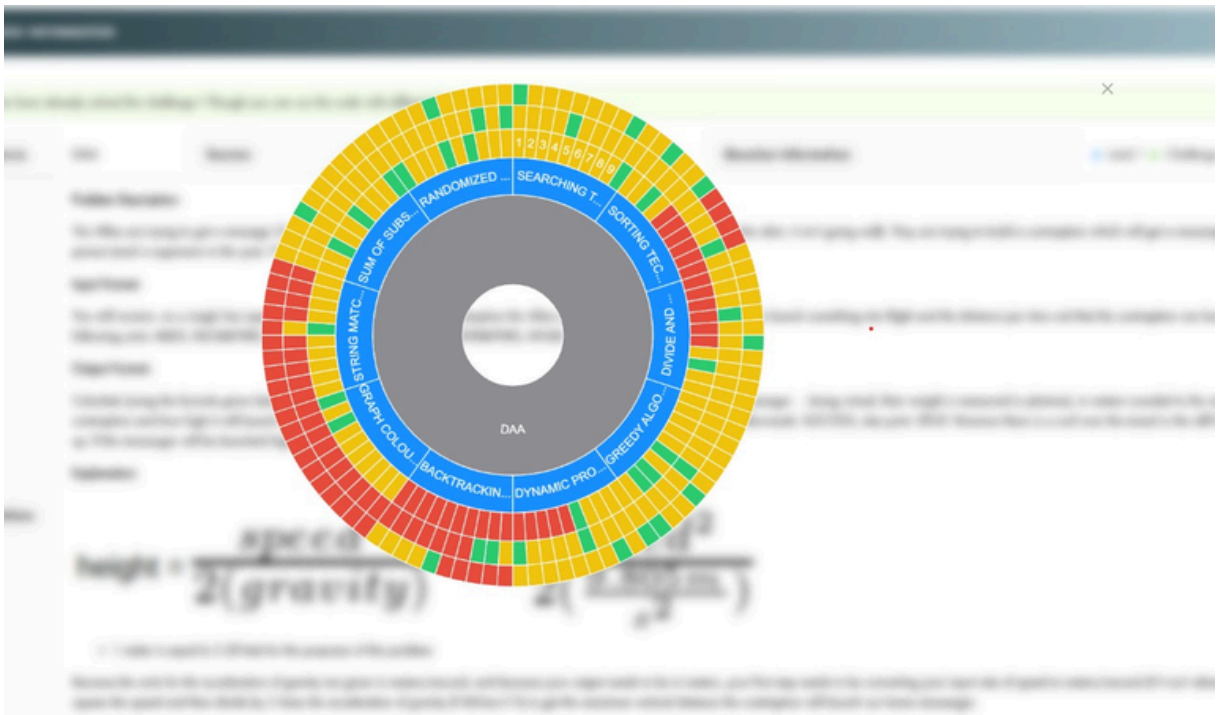
STUDENT PORTFOLIO

Name: B. SAI PRADEEP REDDY
Register Number: RA2311030010315
Mail ID: sr0983@srmist.edu.in
Department: NETWORKING AND COMMUNICATIONS
Semester: IV
Linkedin: https://www.linkedin.com/in/sai-pradeep-reddy-56451934b/
Github:https://github.com/saipradeep123-star

Subject Title: 21CSC204J Design and Analysis of Algorithm

Handled By: Dr. M.Jeyaselvi

E-Lab Completion Status



Explanation of one program

```
#include<bits/stdc++.h>
using namespace std;
int main(){
int n,k;cin>>n>>k;
long r=0;
for(int i=1;i<=min(k,n/2);++i) r+=2*(n-2*i+1)-1;
cout<<r;

}
```

Explanation

1. **#include<bits/stdc++.h>**
   This is a header file that includes all standard C++ libraries. Useful for competitive programming.

2. **using namespace std;**
   This allows you to use standard library functions and types without the std:: prefix.

3. **int main(){ int n, k; cin >> n >> k;**

   You take two inputs from the user:

   - n: Total number of elements (or size of a structure).
   - k: Maximum number of operations/selections allowed.

4. **long r = 0;**
   This is the variable where the final result is accumulated.

5. **for(int i = 1; i <= min(k, n/2); ++i)**

   You iterate i from 1 to the minimum of k and n/2.

   This ensures:

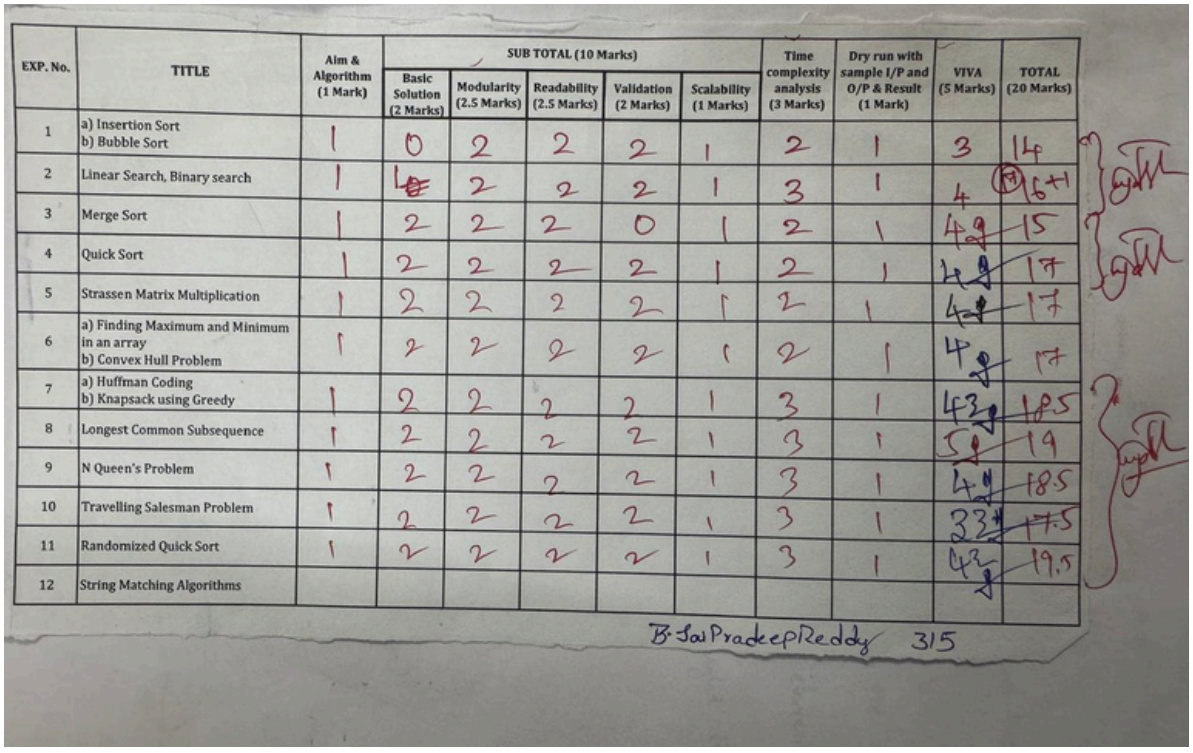
   - You don't exceed the number of allowed operations k.
   - You don't form more than n/2 pairs, which would be invalid in case of pairwise logic.

6. **r += 2 * (n - 2 * i + 1) - 1;**
   In each iteration, you calculate a value and add it to r.

Let's simplify this expression:
Value added=2 · (n−2i+1)−1=2n−4i+2−1=2n−4i+1\text{Value added} = 2 \cdot (n - 2i + 1) - 1 = 2n - 4i + 2 - 1 = 2n - 4i + 1Value added=2 · (n−2i+1)−1=2n−4i+2−1=2n−4i+1
So, on each iteration, you are adding a decreasing value to r.

7. **cout << r;**
   Finally, you print the accumulated result.

Lab Experiment Completion status



EXP. No.	TITLE	Aim & Algorithm (1 Mark)	SUB TOTAL (10 Marks)					Time complexity analysis (3 Marks)	Dry run with sample I/P and O/P & Result (1 Mark)	VIVA (5 Marks)	TOTAL (20 Marks)
			Basic Solution (2 Marks)	Modularity (2.5 Marks)	Readability (2.5 Marks)	Validation (2 Marks)	Scalability (1 Marks)				
1	a) Insertion Sort b) Bubble Sort	1	0	2	2	2	1	2	1	3	14
2	Linear Search, Binary search	1	1	2	2	2	1	3	1	4	16+1
3	Merge Sort	1	2	2	2	0	1	2	1	4	15
4	Quick Sort	1	2	2	2	2	1	2	1	4	17
5	Strassen Matrix Multiplication	1	2	2	2	2	1	2	1	4	17
6	a) Finding Maximum and Minimum in an array b) Convex Hull Problem	1	2	2	2	2	1	2	1	4	17
7	a) Huffman Coding b) Knapsack using Greedy	1	2	2	2	2	1	3	1	4	18.5
8	Longest Common Subsequence	1	2	2	2	2	1	3	1	5	19
9	N Queen's Problem	1	2	2	2	2	1	3	1	4	18.5
10	Travelling Salesman Problem	1	2	2	2	2	1	3	1	3	17.5
11	Randomized Quick Sort	1	2	2	2	2	1	3	1	4	19.5
12	String Matching Algorithms										

B. Sai Pradeep Reddy 315

REAL WORLD APPLICATION IN DAA PPT VR/SIMULATION DEMO

Problem Selection & Description

- **Selecting the best combination of items within a fixed budget.**
- **Useful for shopping, packing, and stock investment.**
- **Goal: Maximize value without exceeding budget.**

Why Knapsack for Budget Shopping?

- **Real-World Budgeting Challenges**
- Limited budget but multiple items to choose from.
- People often struggle to optimize spending for maximum benefit.
- **How the Knapsack Algorithm Helps**
- Finds the best combination of items without exceeding budget.
- Ensures maximum value for money.
- Can be applied to groceries, electronics, and even event planning.

Time Complexity Analysis

- **Brute Force: O(2^n) (Exponential, not practical).**
- **Dynamic Programming: O(nW) (Efficient for real-world use).**
- **Ensures the optimal selection for a given budget.**

REAL WORLD APPLICATION IN DAA PPT VR/SIMULATION DEMO

```c
#include <stdio.h>

// Function to find the maximum value in the knapsack
int knapsack(int weights[], int values[], int n, int capacity) {
    // Create a 2D DP table with (n+1) rows and (capacity+1) columns
    int dp[n+1][capacity+1];

    // Build the table in a bottom-up manner
    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= capacity; w++) {
            if (i == 0 || w == 0) {
                dp[i][w] = 0; // Base case: no items or zero capacity
            } else if (weights[i-1] <= w) {
                // If the item can be included, take the maximum of:
                // 1. Not including the item
                // 2. Including the item
                dp[i][w] = (dp[i-1][w] > dp[i-1][w-weights[i-1]] + values[i-1])
                           ? dp[i-1][w]
                           : dp[i-1][w-weights[i-1]] + values[i-1];
            } else {
                dp[i][w] = dp[i-1][w]; // Item cannot be included
            }
        }
    }
```

```c
    // The last cell contains the maximum value
    return dp[n][capacity];
}

// Example usage
int main() {
    int weights[] = {2, 3, 4, 5};  // Weights of items
    int values[] = {3, 4, 5, 6};   // Corresponding values of items
    int capacity = 5;              // Maximum capacity of the knapsack
    int n = sizeof(weights) / sizeof(weights[0]);  // Number of items

    // Call the knapsack function
    int maxValue = knapsack(weights, values, n, capacity);

    printf("Maximum value in Knapsack: %d\n", maxValue);

    return 0;
}
```

For the given input:

c — Copy — Edit

```c
int weights[] = {2, 3, 4, 5};
int values[] = {3, 4, 5, 6};
int capacity = 5;
```

The output will be:

yaml — Copy — Edit

```yaml
Maximum value in Knapsack: 7
```

CODE FORCES

CODEFORCES
Sponsored by TON

reddysaipradeep7 | Logout

HOME TOP CATALOG CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP

MAIN ACMSGURU | PROBLEMS SUBMIT STATUS STANDINGS CUSTOM TEST

my only friends only

Contest status

#	When	Who	Problem	Lang	Verdict	Time	Memory
318390668	May/05/2025 08:18 UTC+5.5	reddysaipradeep7	112A - Petya and Strings	C++17 (GCC 7-32)	Accepted	124 ms	100 KB
318390617	May/05/2025 08:17 UTC+5.5	reddysaipradeep7	4A - Watermelon	C++17 (GCC 7-32)	Accepted	92 ms	100 KB
318390602	May/05/2025 08:16 UTC+5.5	reddysaipradeep7	110A - Nearly Lucky Number	C++17 (GCC 7-32)	Accepted	154 ms	100 KB
318390578	May/05/2025 08:16 UTC+5.5	reddysaipradeep7	1030A - In Search of an Easy Problem	C++17 (GCC 7-32)	Accepted	61 ms	100 KB
318390551	May/05/2025 08:15 UTC+5.5	reddysaipradeep7	1A - Theatre Square	C++17 (GCC 7-32)	Accepted	62 ms	100 KB
318390502	May/05/2025 08:14 UTC+5.5	reddysaipradeep7	158A - Next Round	C++17 (GCC 7-32)	Accepted	124 ms	100 KB
318390492	May/05/2025 08:14 UTC+5.5	reddysaipradeep7	617A - Elephant	C++17 (GCC 7-32)	Accepted	46 ms	100 KB
318390430	May/05/2025 08:12 UTC+5.5	reddysaipradeep7	1080B - Margarite and the best present	C++17 (GCC 7-32)	Accepted	61 ms	100 KB
318390389	May/05/2025 08:11 UTC+5.5	reddysaipradeep7	1097A - Gennady and a Card Game	C++17 (GCC 7-32)	Accepted	46 ms	100 KB
318390326	May/05/2025 08:10 UTC+5.5	reddysaipradeep7	1011A - Stages	C++17 (GCC 7-32)	Accepted	62 ms	100 KB

Sort by: Default order Submission time Judging Time Solution Size Execution Time

← 1 2 3 4 5 →

SIGNATURE