Activity Selection using Greedy Approach:

Given a set of activities with start and finish times, the greedy approach selects activities that don't overlap, aiming to maximize the number of activities performed.

Algorithm:

1. Sort activities based on finish times.
2. Select the first activity.
3. For each remaining activity, if its start time is after or equal to the previous activity's finish time, select it.

Code:

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Activity {
   int start, finish;
};

bool comparison(Activity a, Activity b) {
   return a.finish < b.finish;
}

void printMaxActivities(Activity activities[], int n) {
   sort(activities, activities + n, comparison);
   cout << "Selected Activities: ";
   int i = 0;
   cout << "(" << activities[i].start << ", " << activities[i].finish << ") ";
   for (int j = 1; j < n; j++) {
      if (activities[j].start >= activities[i].finish) {
         cout << "(" << activities[j].start << ", " << activities[j].finish << ") ";
         i = j;
      }
   }
}

int main() {
   Activity activities[] = {{1, 2}, {3, 4}, {0, 6}, {5, 7}, {8, 9}, {5, 9}};
   int n = sizeof(activities) / sizeof(activities[0]);
   printMaxActivities(activities, n);
   return 0;
}
```

Output:

```
Selected Activities: (1, 2) (3, 4) (5, 7) (8, 9)
```

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____