Subset Sum
Team Cthulu

Tushar Iyer & Aziel Shaw

Rochester Institute of Technology

7th November, 2018
Outline

1. Introduction
   - Summary

2. Sequential Program
   - Sequential Program - Design & Operation

3. Parallel Program
   - Parallel Program - Design & Operation

4. Demonstration
   - Project Demonstration

5. Conclusion
   - Final Remarks
   - Questions
Summary - Subset Sum

- Is an NP-Complete problem.
- Given a set of numbers and a target sum, find a set that totals that target sum.
- Dynamic Programming algorithm achieves best results both for performance and parallelization.
Algorithm 1 SubsetSumSeq

**Ensure:** A subset which adds up to a given sum.

1: **procedure** SubsetSeq(Set, Sum)
2:    for index 0 to n do
3:        dynArray[0][index] = TRUE
4:    for index 1 to sum do
5:        dynArray[index][0] = FALSE
6:    for index 1 to sum do
7:        for index2 1 to length(Set) do
8:            dynArray[index][index2] = dynArray[index][index2 - 1]
9:        if index >= Set[index2 - 1] then
10:           dynArray[index][index2] = dynArray[index][index2]
11:        or
12:           dynArray[index][index2] = dynArray[index2 - 1][index2 - 1]
Sequential Program - Design & Operation

1. Init
2. Read input string and instantiate SubsetSpec object
3. Verify that Set is valid
   - Pass
   - Fail
4. Initialize 2D dynArray and set table to base cases
5. Fill table in bottom-up fashion
6. Check if dynArray(sum - 1)[length - 1] = T
   - True: Set found
   - False: Set not found
   - End

Options for Set creation:
- RandomSet(lb, ub, seed, size, sum)
- SameSet(num, size, sum)
- LinearSet(start, step, size, sum)
- FibonacciSet(size, sum)
Algorithm 2 SubsetSumSmp

**Ensure:** A subset which adds up to a given sum.

1: **procedure** SubsetSmp(Set, Sum)
2: **parallelFor** index 0 to n do
3:   dynArray[0][index] = TRUE
4: end
5: **parallelFor** index 1 to sum do
6:   dynArray[index][0] = FALSE
7: end
8: **parallelFor** index 1 to sum do
9:   **for** index2 1 to length(Set) do
10:   dynArray[index][index2] = dynArray[index][index2 - 1]
11:   if index >= Set[index2 - 1] then
12:     dynArray[index][index2] = dynArray[index][index2]
13:   or
14:     dynArray[index][index2] = dynArray[index2 - 1][index2 - 1]
15: end
Parallel Program - Design & Operation

1. **Init**
   - RandomSet(lb, ub, seed, size, sum)
   - SameSet(num, size, sum)
   - LinearSet(start, step, size, sum)
   - FibonacciSet(size, sum)

2. **Read input string and instantiate SubsetSpec object**

3. **Verify that Set is valid**
   - **Pass**
     - Initialize 2D dynArray and set table to base cases
     - N threads

4. **Fill table in bottom-up fashion**
   - N threads

5. **Check if dynArray(sum - 1)[length - 1] = T**
   - **True**
     - Set found
   - **False**
     - Set not found

6. **End**
Demo Time
Final Remarks

- From our tests it is apparent our initial thoughts were correct
  - Parallelization helps, but only to a point
  - We believe we are seeing diminishing returns
  - Time increases as the size of the set increases

- We are looking forward to graphing the results
Any Questions?