Subset Sum Problem

Caleb Lundquist & Josh Higham

Introduction

- Computer Passwords
  - Instead of storing a password, a computer can store a specific sum of a subset on a file.
- Message Verification
  - Computers can check for an exact subset sum to ensure the returned message is from the proper source.

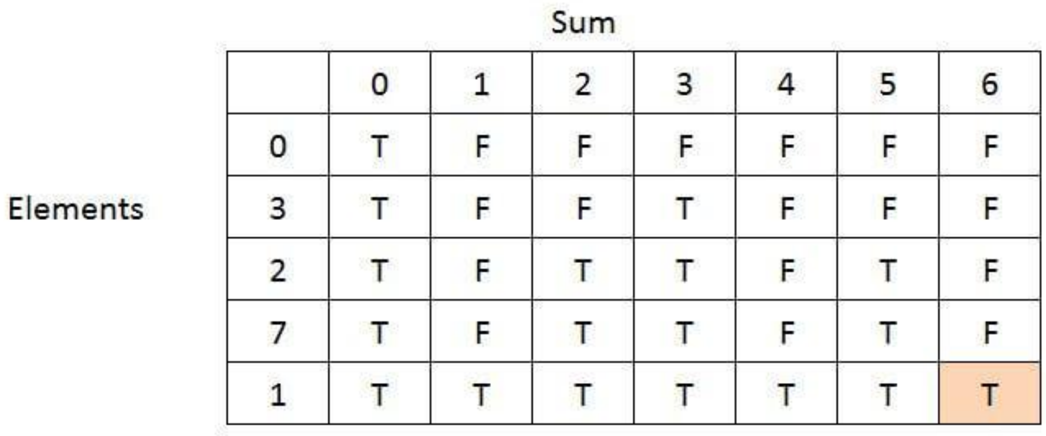
- NP-Complete
  - Dynamic Programming
  - Parallelization
- N := the number of integers in the set.
- Target := the target sum of a potential subset.

Initialization

- Process 0 creates Int_Set and populates it with random integers in [0, 2N]
- Process 0 creates Target in [.5*N^2, N^2]
- Process 0 sends Int_Set and Target to all other Processes
- Each process initializes Cache
    - Cache is a 2D Array size [N][Target+1]
    - -0th column is all True (any set can have a subset sum of 0 due to empty set)
    - -0th row is all false except 0 and Int_Set[0] columns (set of 1 int can make sum 0 and only value)

Dynamic Programming

- Cache must be computed row by row.
- Each process computes columns [Rank*(x),Rank*(x)+x] where x := Target / MPI_Size
- Upon completion, Process 0 returns results.



Sum

Elements

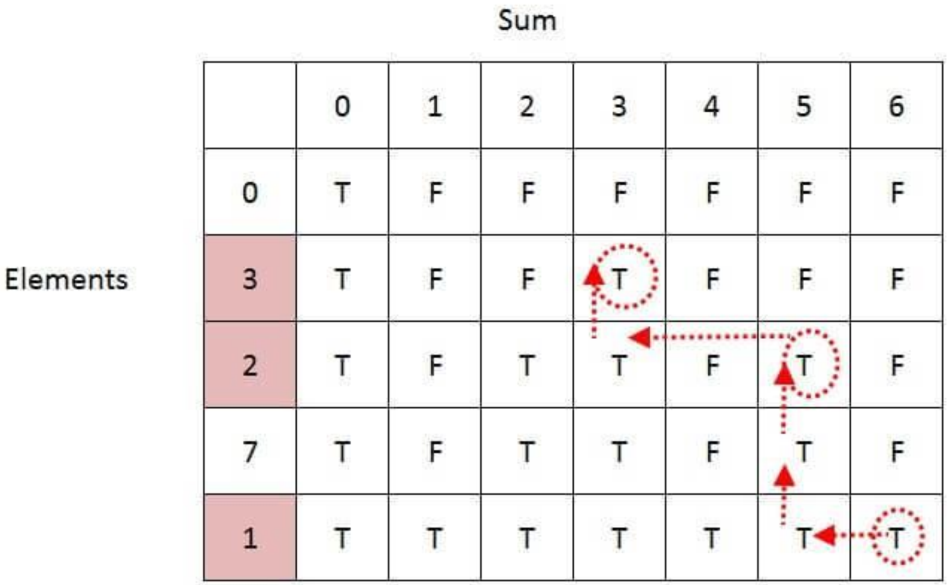
	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
2	T	F	T	T	F	T	F
7	T	F	T	T	F	T	F
1	T	T	T	T	T	T	T

Parallelization

- On each row, once the processes are done computing, they must share their results.
- Each process fills in the Cache with the results from the other processes.
- The Cache on every process must by up to date and identical to the others before proceeding to compute next row.

Traceback

- Which subset produced our target sum?



Sum

Elements

	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
2	T	F	T	T	F	T	F
7	T	F	T	T	F	T	F
1	T	T	T	T	T	T	T

Include the current element whenever you move left.

Closest Sum

- If there is no subset with sum == Target, what is the closest we could get?

Conclusion

- As long as the target sum is large, the benefit is by a factor of the number of processes.
- MPI Limitations cause freezing when arrays get too big to send.