COMP90025 Parallel and Multicore Computing

Project 1B - Knapsack filling

Aaron Harwood and Lachlan Andrew

School of Computing and Information Systems
The University of Melbourne

2019 Semester II

Problem Summary

- The *knapsack* problem is a classic NP-hard resource allocation problem, which is used in the management of parallel systems.
- You are given a knapsack that is capable of holding up to a fixed weight called its capacity, C.
- You are also given a collection of items, each of which has a weight w_i and a value v_i.
- The task is to choose which items to put into the knapsack to maximize the *value* of its contents.
- Being NP-hard means that there is (probably) no algorithm that can solve the problem time polynomial in the number of bits of its input.
- However, there is a "pseudo-polynomial time" solution: one that is polynomial in the largest value in the input, namely the capacity.
- This algorithm is based on dynamic programming.
- It sequentially finds the best solution for $C' = 1$, for $C' = 2$ etc., up to $C' = C$.

Code Summary and Input Format

- A sequential algorithm for solving the integer knapsack problem has been provided on LMS. The program reads an instance from standard input (`stdin`) and outputs the maximum value that the knapsack can carry.
- The first line of input is a single integer (`long int`) giving the capacity of the knapsack.
- The second line is a single integer (`int`) giving the number of items that may be placed into the knapsack.
- Each subsequent line of the input file, one per item, provides a pair of positive integers:

  value weight

(Tiny) Example Input

The following example can be solved sequentially in microseconds. Your submission will be tested on examples that take minutes to solve sequentially.

```
100
5
83 83
62 62
43 43
20 20
5 5
```

Tasks

- You can either work individually, or with the same partner as for Part A. If there is a compelling reason to change partners, please contact Lachlan Andrew.
- Write an MPI program that computes and outputs the maximum value that can be carried in the knapsack, in the same format as done by the sequential program.
- Aim to have your program run as fast as possible. In doing so, you may alter the calculations of the program, so long as the final output is correct. You can use *algorithms* that you find in the literature (parallel or sequential), but please do not directly copy code from elsewhere.
- Use the sequential code as a base. Do not change the file before the end of `main()`. If you need to include additional headers, they can be included after `main()`. In particular, do not alter the timing statements or add any of your own timing statements.
- Write at most 750 words that outlines how you achieved parallelism/high performance. Include tables and/or charts of your own measurements that support your discussion.

Assessment

- Project 1 B is worth **8%** of your total assessment. It is group work in size of 2 at most.
- Assessment of the report (3/8) is based on the level of details and presentation.
- Assessment of the program (5/8) is based on correctness and performance. Incorrect programs (i.e. that give incorrect outputs or that fail to compile/run) will attract few if any marks. The top 5 fastest running programs, when given a mystery work load (you will not be told the work load in advance), will be given a bonus mark; i.e. the maximum mark for this project part is 9/8.
- At most five people will get the bonus mark. If a sixth person is tied with anyone in the top five, then anyone tied with that sixth person will not receive a bonus mark.

Submission

- Submit a PDF of your report (use PDF only, no other format will be assessed) via LMS on or before **Saturday 14th September**. As well you will need to submit your program directly on Spartan. The process will be similar to that for Project 1a, but instructions for doing this will be given closer to the deadline.
- Optimal code often requires specific compiler options. As a comment in the last line of your C code, put the exact command line used to compile your code. Do not put anything else on that line
- Use 10pt font, single line spacing, 25 mm margins all around and double column formatting. Use appropriate headings and clearly label and refer to tables, figures and references.
- Clearly put your name and login name and those of your partner at the top of the report.