Semestrálna práca MI-PAR 2013/2014:

Paralelný algoritmus pre rozklad obdĺžnikov

Jakub Melezínek

Martin Klepáč

November 22, 2013

1 Definícia problému

Našou úlohou bolo vytvoriť program, ktorý implementuje usporiadanie obdĺžnikov v 2D mriežke so zachovaním minimálneho celkového obvodu týchto obdĺžnikov.

Obsah obdĺžnika je daný hodnotou uloženou v 2D mriežke, pričom tento element musí byť súčasťou obdĺžnika s daným obsahom.

Jednotlivé obdĺžniky sú vzájomne disjunktné až na spoločné vrcholy a hrany, pričom zjednotenie všetkých obdĺžnikov pokrýva pôvodnú mriežku.

Riešení, ako rozdeliť mriežku na jednotlivé obdĺžniky, môže byť viac – v takom prípade hľadáme riešenie s minimálnym obvodom – zároveň ale riešenie nemusí existovať.

2 Formát vstupu, výstupu

Formálne, vstup vyjadríme pomocou

- a, b = prirodzené čísla predstavujúce rozmery mriežky
- $H[1..a][1..b]$ = mriežka
- n = prirodzené číslo predstavujúce počet obdĺžnikov vo vnútri mriežky

Výstupom algoritmu je okrem celkového obvodu dielčích obdĺžnikov vyfarbená mriežka, t.j. mriežka, v ktorej každému elementu je priradené písmeno abecedy, ktoré jednoznačne identifikuje obdĺžnik, ktorého je bod súčasťou.

3 Implementácia sekvenčného riešenia

Primárny cieľ sekvenčného riešenia, na ktorom ďalej staviame v paralelnej implementácii, spočíva v nájdení a následnom prehľadaní celého stavového priestoru množiny potenciálnych riešení.

Veľkosť mriežky	T(n) [s]
15x15	29
20x20	473
21x21	571
23x23	983

Table 1: Trvanie sekvenčného výpočtu

Stavový priestor v našom prípade rozumieme množinu mriežok, v ktorých postupne spracúvame obdĺžniky s dvojicou parametrov

1. *shape* - veľkosť obdĺžnika (dĺžka x šírka)
2. *position* - pozícia v mriežke

Očividne, *shape* obdĺžnika je určený jeho plochou, zatiaľ čo *position* je vlastnosťou mriežky a ostatných obdĺžnikov v nej existujúcich - napr. obdĺžnik nedokážem vložiť do mriežky, pokiaľ jeden z jeho rozmerov presahuje veľkosť mriežky alebo je okolie obdĺžnika posiate inými, už zafixovanými obdĺžnikmi.

Popis sekvečného riešenia v skratke: procesor zo zásobníka mriežok vezme obdĺžnik, zistí, či má *shape* (v prípade negatívnej odpovede vygeneruje všetky dvojice *a*, *b* tak, aby ich súčin bol rovný očakávanej ploche). V ďalšom kroku procesor zistí, či daný obdĺžnik má stanovenú *position* v mriežke - ak nie, na zásobník uloží všetky prípustné možnosti s ohľadom na veľkosť mriežky a susedné obdĺžniky. Procesor pokračuje do spracovania posledného obdĺžnika alebo do nájdenia prvého obdĺžnika, ktorý nemožno umiestniť do mriežky. V prípade, ak sa všetky obdĺžniky podarilo umiestniť do mriežky, procesor porovná výsledok s doterajším minimom a zo zásobníka vezme ďalšiu mriežku a opakuje takto popísanú akciu.

V nami implementovanom triviálnom sekvenčom riešení neuplatňujeme orezávanie neperspektívnych ciest - každá mriežka dobehne do konca v prípade existencie rozdelenia obdĺžnikov a až následne sa porovná výsledný obvod s doterajším minimom.

Trvanie sekvenčného výpočtu na výpočtovom klastri star.fit.cvut.cz je znázornené v tabuľke 1. Výsledný sekvenčný čas je určený ako aritmetický priemer trojice meraní.

4 Príklad zadania a výsledku

Program spúšťame s parametrom -f, ktorý udáva cestu k súboru obsahujúce vstupné dáta. Pre triviálny vstup o veľkosti mriežky 5x5 dostávame výstup zobrazený na obrázku 1.

5 Implementácia paralelného riešenia

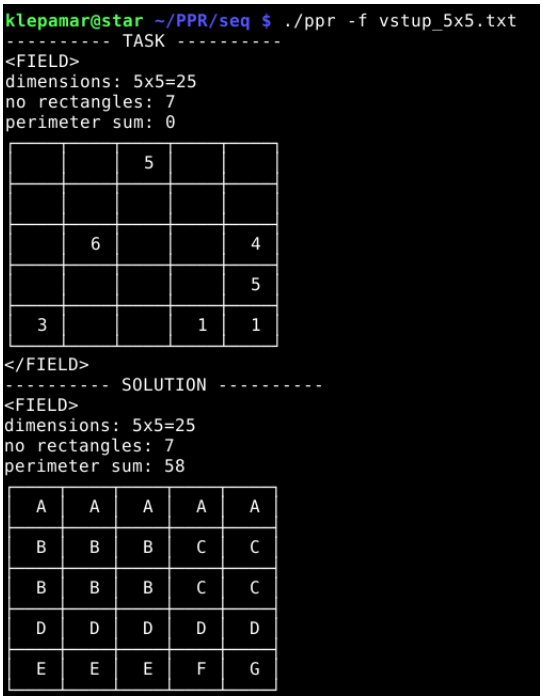

Figure 1: Príklad výstupu pre sekvenčnú úlohu

Veľkosť mriežky	**T(n,2)**	**T(n,4)**	**T(n,8)**	**T(n,16)**	**T(n,24)**	**T(n,32)**
15x15						
20x20						
21x21						
23x23						

Table 2: Trvanie paralelného výpočtu