Párhuzamos programozás

Rendezések

Készítette: Györkő Péter
EHA: GYPMABT.ELTE
Nappali tagozat
Programtervező matematikus szak

Budapest, 2009 május 9.

Bevezetés

A számítástechnikában felmerülő problémák jelentős részében szükség van elemek rendezésére, sorbaállítására. Ehhez definiálni kell egy rendezési relációt, mely bármely két elemet össze tud hasonlítani. Ehhez egész számok esetében általában a hagyományos relációs jeleket szokás használni, így a feladat az, hogy növekvő, vagy csökkenő sorozatba állítsuk az elemeket.

A soros rendező algoritmusok két csoportba oszthatók: az összehasonlításon alapulókra és a hasítófüggvényt használókra. Írásomban az első csoporttal foglalkozom. Mivel az azonos elemek a rendezés számára érdektelenek, így az általánosság megszorítása nélkül feltehetem, hogy az összes elem különbözik. Szintén feltehető, hogy amennyiben n elemet kívánunk rendezni, úgy ezen elemek legyenek az 1-től n-ig lévő egész számok egy permutációja. Az egyszerűség kedvéért példáimban az n kettőhatvány (felírható 2^k alakban, ahol k természetes szám) lesz, így egyszerűbbek lesznek a számolások és az ábrák. Amennyiben a rendezendő elemek száma nem ilyen, úgy válasszuk ki a legkisebb olyan kettőhatványt, amely nem kisebb a kérdéses számnál, majd a maradék helyeket töltsük ki tetszőlegesen nagy vagy kicsi számokkal.

Az összehasonlításon alapuló rendezéseknél a műveletigényt úgy határozzuk meg, hogy az elvégzendő összehasonlítások számával arányosítjuk a futásidőt, és az egyéb műveleteket (például az elemek mozgatása, törlése, indexelése) a hatékonysági vizsgálatok esetében figyelmen kívül hagyjuk.

A továbbiakban a lg (logaritmus) függvényt kettes alapúnak tekintem, mivel a számítástechnika és a kettes számrendszer használata mellett ez a legkézenfekvőbb.

A soros rendezés

Az összehasonlító rendezések alaptétele kimondja, hogy nem létezik aszimptotikusan gyorsabb rendezés mint n*lg(n) (ahol n a rendezendő elemek számát mutatja). Ez a korlát a gyakorlatban elérhető, például a merge sort (összefuttatásos rendezés) segítségével. Ennek alapja egy rekurzív eljárás, mely során kezdetben a szomszédos elemeket “fésüljük össze”, majd a párokat és így tovább a k-adik menetben egymás melletti 2^k elemek kerülnek összefuttatásra.

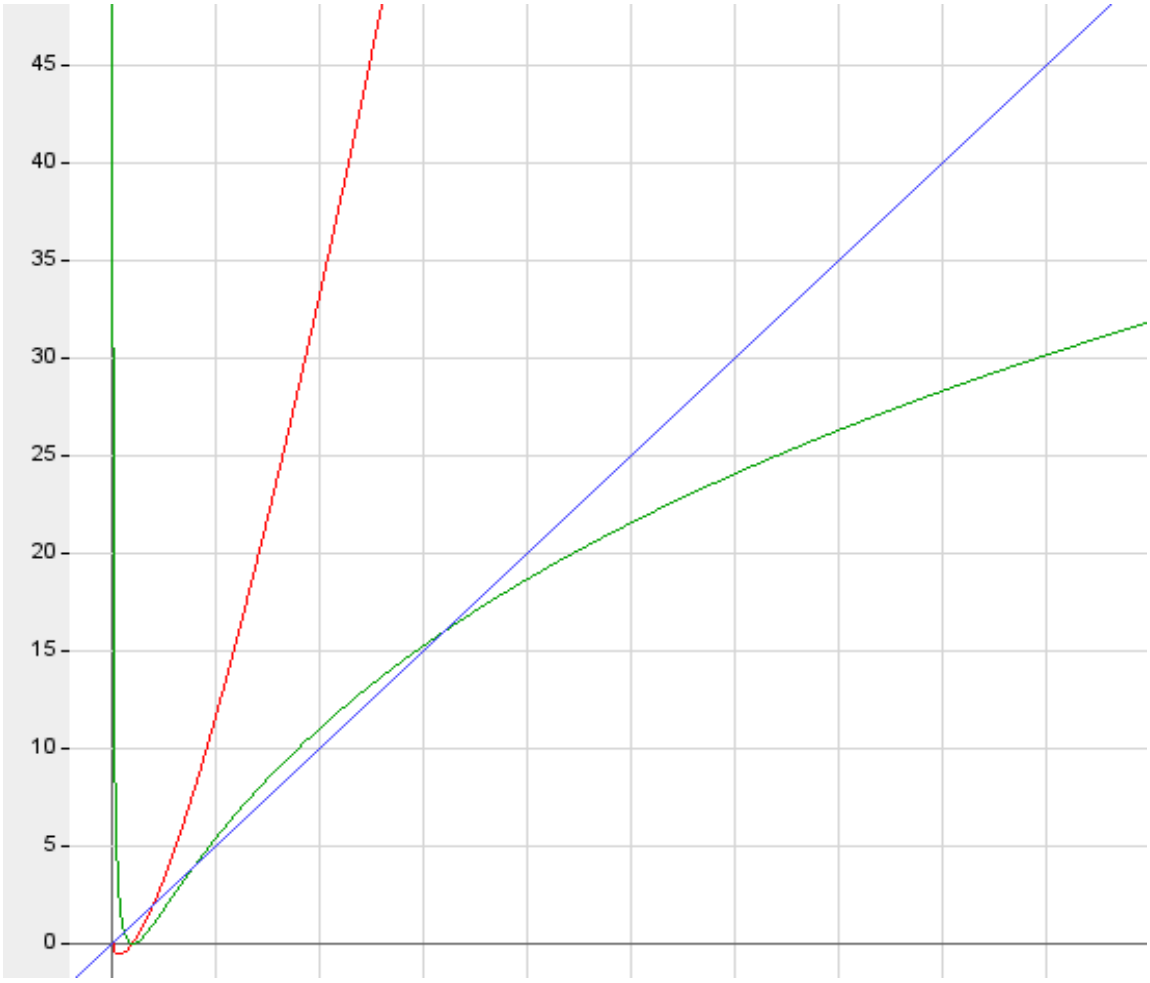
Mivel maga a fésülés művelet lineáris időben elvégezhető (m és n hosszú tömb esetén maximum m+n–1 összehasonlítás elegendő), így minden szinten O(n) összehasonlítás szükséges. A rendezés során kapott fa magassága log(n), így a rendezéshez szükséges idő éppen n*lg(n), vagyis elértük az elméleti határt.

A párhuzamos algoritmus

A következőkben a CREW PRAM model szerinti párhuzamos gépeket fogom tekinteni, mely esetében megengedett egy memóriaterület párhuzamos olvasása, ám írása nem.

Az algoritmus alapja az előzőekben ismertetett merge sort és egy munkaoptiomális összefésülő algoritmus, mely képes O(lg(m)) lépésben összefésülni két m hosszú rendezett sorozatot 2m/lg m processzoron. Ennek elve, hogy mindkét sorozatot m/lg m részre bontjuk, majd mindegyikhez egy processzort állítunk, ami az ismert soros algoritmus segítségével összefésüli azokat.

Ezzel a k-adik szinten mindig 2 darab 2^(k-1) hosszú részeket fésülhetjünk össze párhuzamosan. Ennek műveletigénye az előbbi algoritmust használva lg(2^(k-1)) = k–1, vagyis szintenként csak lineáris munkát kell végeznünk. Ez egy számtani sort alkot, melynek összege k(k-1)/2 ami négyzetes, ám ne feledjük, hogy k=lg(n), vagyis O(lg(n)^2) az algoritmus műveletigénye, mely összehasonlítva a soros esettel elég szembetűnő javulást hoz, melyet az alábbi grafikon jól szemléltet:



*a piros függvény az x*ln(x), a kék pedig az ln(x)^2*
a kek csak a szemléltetés kedvéért szerepelő identitás függvény

Amennyiben a szükséges processzorok számát tekintjük, úgy már nem ilyen kedvező a helyzet. Az első lépésben az összefűzéshez legalább n/2 db processzorra van szükségünk, ám az utolsóban már csak n/lg(n).

Végül egy táblázatban összefoglaltam az általam írt szimulációs program segítségével kapott eredményeket:

n		**Merge Sort**	**Parallel Merge Sort**
8	Összehasonlítások száma	15	5
	Átlagos processzorszám	1	5.6
32	Összehasonlítások száma	123	81
	Átlagos processzorszám	1	15.8
128	Összehasonlítások száma	730	523
	Átlagos processzorszám	1	48.71
512	Összehasonlítások száma	3951	2959
	Átlagos processzorszám	1	159.4
2048	Összehasonlítások száma	19931	15470
	Átlagos processzorszám	1	541.5
8192	Összehasonlítások száma	96103	77002
	Átlagos processzorszám	1	1888

a python nyelven írt program elérhető a http://people.inf.elte.hu/~gy_p/sort.py címen

Felhasznált irodalom

Fekete István - Algoritmusok és adatszerkezetek I. :
http://people.inf.elte.hu/fekete/alg_adat_1_text.html

Iványi Antal – Párhuzamos algoritmusok:
http://elek.inf.elte.hu/magyarkonyvek