簡潔データ構造 第 2 回

marimo

2023 年 4 月 25 日

方針

実は現実のマシンと Word-RAM は全然違う

- メモリは word 単位でしか読めない
  - もっと言うと読むのはライン単位
- 表引きは遅い
  - RAM アクセスは 10ns ぐらいかかるが 1 クロックは 0.5ns ぐらい
- もっと多くのことが 1 命令でできる
  - word 内の rank,select はできる

メモリの読み方

Word-RAM はメモリを任意の bit から $w\,\mathrm{bits}$ だけ読み出すことができた。
実際のプロセッサはメモリはワードサイズ単位とかバイト単位とかでしか読めないので，小ブロックがワードをまたがると実行速度上の問題がある。
一方でバイト単位にすると索引サイズが大きくなるので困る。
紹介した rank の索引では $n = 2^{32}$ だと大ブロックは $l = 2^{10}$ の長さになり，小ブロックの配列に入る整数は $10\,\mathrm{bits}$ で困る。 そこで大ブロックの長さを 2^{16} ぐらいに固定してやると小ブロックで持つ整数が 16bit でちょうど持ててよい。

表引きは遅い

使う表は $\Omega(\sqrt{n}\log\log n)$ ぐらいでかなり大きくなりがちなのでキャッシュに乗らない

RAM にアクセスすることになるがこれは 10ns ぐらいかかる

一方で bit 演算とかは平均すれば 1 命令 0.5 クロックぐらいでできる (およそ 0.3ns) なので, bit 演算で頑張った方が速くなることが多い

表を使わない場合，小ブロックを $\frac{1}{2}\lg n\,\mathrm{bits}$ にする必要がなくなるので小ブロックをもう少し大きくても大丈夫になる。具体的にはワードサイズぐらいか，キャッシュのラインサイズを考慮して 256 ビットぐらい？

CPU の拡張命令

現代の CPU は実は word-size の四則と bit 演算と少しの制御よりももっと沢山のことができる
最近の x64 プロセッサについている拡張命令として, POPCNT と PDEP と TZCNT というのがある

POPCNT x	x(1 ワード) の中の 1 の数を数える
PDEP x y	x の下位 POPCNT y ビットを y の立っているビットのところに分配する
TZCNT x	x の下位の連続する 0 の個数を数える

これを使えば表を使わない方針を高速化できる。

$$\mathrm{rank}(x, i) = \mathrm{POPCNT}(x << w - i)$$
$$\mathrm{select}(x, i) = \mathrm{TZCNT}\big(\mathrm{PDEP}(1 << i, x)\big)$$

CPU の拡張命令

最適化で使ってくれるときもあるが，明示的に使いたい場合は次の関数を使う

	POPCNT	PDEP
gcc	`__builtin_popcount`	`_pdep_u64`
clang	`<x86intrin.h>`内の `_mm_popcnt_u64`	`<x86intrin.h>`内の `_pdep_u64`
rustc	`std::arch::x86_64`内の `_popcnt64`	`std::arch::x86_64`内の `_pdep_u64`

ただし，gcc,clang はコンパイラオプションで `-msse4.2,-mbmi2` を必要とする。また rustc は RUSTFLAG に `-C target-feature=+bmi2,+sse4.2` を付ける必要がある。

selectについて

こちらには rank ほどのめぼしい高速化，簡単化はない
先程述べた，葉で select をするときの高速化ぐらい