Radix sort

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-02-24

Counting sort

Hugmynd: notum gildin sem á að raða sem sætisnúmer í fylki sem telur tíðni þeirra.

Dæmi:

- Röðum: 6,1,4,4,4,2,1
- `count[6]++;`
- `count[1]++;`
- `count[4]++;`
- `count[4]++;`
- `count[4]++;`
- `count[2]++;`
- `count[1]++;`

Counting sort frh.

Dæmi frh.:

- Röðum: 6,1,4,4,4,2,1
- Staðan á count eftir talningu:

```
count[0] = 0
count[1] = 2
count[2] = 1
count[3] = 0
count[4] = 3
count[5] = 0
count[6] = 1
count[7] = 0
count[8] = 0
...
count[max] = 0
```

Counting sort frh.

- Röðum: 6,1,4,4,4,2,1
- Staðan á count eftir talningu:

```
count[0] = 0          count[6] = 1
count[1] = 2          count[7] = 0
count[2] = 1          count[8] = 0
count[3] = 0          ...
count[4] = 3          count[max] = 0
count[5] = 0
```

- Göngum í gegnum count og setjum gildin aftur í upphaflega fylkið í vaxandi röð

```
f[0] = 1      f[3] = 4      f[6] = 6
f[1] = 1      f[4] = 4
f[2] = 2      f[5] = 4
```

- Endum með: 1,1,2,4,4,4,6

Tímaflækja counting sort

Tímaflækjan:

- Tímaflækjan er $O(n)$. Línuleg röðun!
- Hinsvegar, þurfum sæti í count fyrir hvert mögulegt gildi.

Counting sort fyrir heiltölur í Java:

- `int` heiltölur í Java hafa 2^{32} möguleg gildi
- Til að raða `int[]` í Java þarf amk. $2^{32} = 4$ GB af minni.

$\Longrightarrow$ counting sort hentar aðallega ef tölurnar eru á litlu bili

Hvað með að nýta svipaða hugmynd á smærri skala?

- Gætum notað tölustafi sem sætisnúmer og raðað eftir einum tölustaf í einu.

Stöðug röðun

Röðunaraðferð er *stöðug* (e. stable) ef hún viðheldur gildandi röð jafnra staka.

Það er, ef A==B og A er á undan B fyrir röðun þá er A einnig á undan B eftir röðun.

Stöðugleiki fyrir röðunaraðferðir:

- Insertion sort er stöðug
- Selection sort er **ekki** stöðug
- Mergesort er stöðug

LSD radix sort

LSD radix röðunaraðferð raðar með stöðugri aðferð eftir sérhverjum tölustaf frá hægri til vinstri, þ.e. fyrst er raðað eftir síðasta tölustafnum, síðan næstsíðasta, o.s.frv.

Þegar búið er að raða eftir n-ta síðasta tölustafnum þá eru tölurnar í röð séu þær bornar saman á grundvelli síðustu n tölustafanna.

Rökstuðningur: Ef tvær tölur hafa ólíkan n-ta síðasta tölustaf þá eru þær augljóslega í réttri röð eftir röðun á grundvelli þess tölustafs. Ef tölustafurinn er hinn sami þá eru þær einnig í réttri röð því stöðugleikinn tryggir að fyrri röðun á grundvelli síðustu (n-1)-tölustafa var viðhaldið.

Tölustafir

```c
int main(void)
{
    int a = 123;
    int v = 1;
    int n;

    for (n = 0; n < 3; n++)
    {
        // v = 10^n
        // Búið er að prenta út síðustu n tölustafi
        // tölunnar a.
        // 0 <= n <= 3.
        int ri = (a/v) % 10;
        printf("%d\n", ri);
        v *= 10;
    }
}
```

Þversumma

```
// Notkum: n = digitsum(x);
// Fyrir:  Ekkert
// Eftir:  n er þversumma x
int digitsum(int x)
{
    int sum = 0;

    while (x)
    {
        sum += x % 10;
        x = x/10;
    }

    return sum;
}
```

```
digitsum(0) = 0
digitsum(100) = 1
digitsum(101) = 2
digitsum(123) = 6
```

Dæmi 2 í V7: LSD radix sort með hjálp biðraða

Reikniritið:

- Við notum 10 biðraðir til að raða eftir tilteknum tölustaf, köllum þær r[0], r[1], ..., r[9]
- Aðalbiðröðin (viðfangið) er tæmd yfir í biðraðirnar 10 eftir gildi tölustafsins sem er undir skoðun, 0 fer í r[0], 1 fer í r[1], o.s.frv.
- Þegar aðalbiðröðin er orðin tóm, þá innihalda r[0], ..., r[9] samanlagt öll upphaflegu gildin
- Í hverri biðröð þá eru tölurnar í vaxandi röð á grundvelli síðustu N tölustafa
- Mokum úr r[0], ..., r[9] aftur yfir í aðalbiðröðina í röðinni r[0] fyrst, svo r[1], o.s.frv.
- Þá eru r[0], .., r[9] tómar og aðalbiðröðin inniheldur tölurnar í vaxandi röð á grundvelli síðustu N+1 tölustafanna

Dæmi

Tökum inn biðröðina `q: 14, 20, 12`

Í fyrstu umferð þá:

- Notum 10 tómar biðraðir til að raða síðasta tölustafnum
- Mokum úr q yfir í r biðraðirnar, þá verður staðan:

```
q: empty
r[0]: 20
r[1]: empty
r[2]: 12
r[3]: empty
r[4]: 14
r[5], .., r[9]: empty
```

- Mokum síðan úr r[0], ..., r[9] yfir í q í röðinni r[0], r[1], ..:

```
q: 20, 12, 14
r[0], .., r[9]: empty
```

Dæmi frh.

Í annarri umferð þá:

- Notum 10 tómar biðraðir til að raða næstsíðasta tölustafnum
- Mokum úr q yfir í r biðraðirnar, þá verður staðan:

```
q: empty
r[0]: empty
r[1]: 12, 14
r[2]: 20
r[3], .., r[9]: empty
```

- Mokum síðan úr r[0], ..., r[9] yfir í q í röðinni r[0], r[1], ..:

```
q: 12, 14, 20
r[0], .., r[9]: empty
```

- Búið að koma q í röð skv. næstsíðasta tölustafnum
- Fyrir jöfn gildi (12 og 14) þá gildir fyrri röð því röðunin er stöðug. Meginforsendan í radix sort.

Tímaflækja

Tímaflækjan:

- n er fjöldi talna í biðröðinni sem á að raða
- Moksturinn úr q í r er $O(n)$ m.v. O(1) enqueue og dequeue
- Moksturinn úr r[0], .., r[9] yfir í q aftur er einnig $O(n)$
- Förum eina umferð í gegnum allar tölurnar fyrir hvern tölustaf

- Ef hámarksfjöldi tölustafa er m þá er heildar tímaflækjan $O(nm)$
- Hámarksfjöldi tölustafa í 32-bita heiltölum er $m = \log_{10}(2^{32})$