

Innbyggðar gagnagrindur

Bergur Snorrason

January 15, 2024

Innbyggðar gagnagrindur í C++

- ▶ Grunnur C++ býr yfir mörgum öflugum gagnagrindum.
- ▶ Skoðum helstu slíku gagnagrindur og tímaflækjur mikilvægust aðgerða þeirra.
- ▶ Við munum bara fjalla um gagnagrindurnar í grófum dráttum.
- ▶ Það er hægt að finna ítarlegra efni og dæmi um notkun á netinu.

Fylki

- ▶ Lýkt og í mörgum öðrum forritunarmálum eru fylki í `C++`.
- ▶ Fylki geyma gögn og eru af fastri stærð.
- ▶ Þar sem þau eru af fastri stærð má gefa þeim tileinkað, aðliggjandi svæði í minni.
- ▶ Þetta leyfir manni að vísa í fylkið í $\mathcal{O}(1)$.

Aðgerð	Tímaflækja
Lesi eða skrifa ótiltekið stak	$\mathcal{O}(1)$
Bæta staki aftast	$\mathcal{O}(n)$
Skeyta saman tveimur	$\mathcal{O}(n)$

vector

- ▶ Gagnagrindin `vector` er að mestu leiti eins og fylki.
- ▶ Það má þó bæta stökum aftan á `vector` í $\mathcal{O}(1)$.
- ▶ Margir nota bara `vector` og aldrei fylki sem slík.

Aðgerð	Tímaflækja
Lesi eða skrifa ótiltekið stak	$\mathcal{O}(1)$
Bæta staki aftast	$\mathcal{O}(1)$
Skeyta saman tveimur	$\mathcal{O}(n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     srand(time(NULL));
7     int i, n = 10;
8     vector<int> a;
9     for (i = 0; i < n; i++) a.push_back(rand()%100);
10    for (i = 0; i < n; i++) printf("%2d ", a[i]);
11    printf("\n");
12    sort(a.begin(), a.end());
13    for (i = 0; i < n; i++) printf("%2d ", a[i]);
14    printf("\n");
15    sort(a.rbegin(), a.rend());
16    for (i = 0; i < n; i++) printf("%2d ", a[i]);
17    printf("\n");
18    return 0;
19 }

```

```

1 >> ./a.out
2 83 23 26 24 52 74 0 39 0 90
3 0 0 23 24 26 39 52 74 83 90
4 90 83 74 52 39 26 24 23 0 0

```

list

- ▶ Gagnagrindin `list` geymir gögn líkt og fylki gera, en stökin eru ekki aðliggjandi í minni.
- ▶ Því er uppfletting ekki hröð.
- ▶ Aftur á móti er hægt að gera smávægilegar breytingar á `list` sem er ekki hægt að gera á fylkjum.

Aðgerð	Tímaflækja
Finna stak	$O(n)$
Bæta staki aftast	$O(1)$
Bæta staki fremst	$O(1)$
Bæta staki fyrir aftan tiltekið stak	$O(1)$
Bæta staki fyrir framan tiltekið stak	$O(1)$
Skeyta saman tveimur	$O(1)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     int i;
7     list<int> a;
8     for (i = 1; i <= 5; i++) a.push_back(i);
9     for (i = 1; i <= 5; i++) a.push_front(i);
10    for (auto p : a) printf("%d ", p);
11    printf("\n");
12    return 0;
13 }

```

```

1 >> ./a.out
2 5 4 3 2 1 1 2 3 4 5

```

stack

- ▶ Gagnagrindin `stack` geymir gögn og leyfir aðgang að síðasta staki sem var bætt við.

Aðgerð	Tímaflækja
Bæta við staki	$O(1)$
Lesi nýjasta stakið	$O(1)$
Fjarlægja nýjasta stakið	$O(1)$


```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     int i, n = 10;
7     stack<int> a;
8     for (i = 0; i < 10; i++) a.push(i);
9     while (a.size() > 0)
10    {
11        printf("%d ", a.top());
12        a.pop();
13    }
14    printf("\n");
15    return 0;
16 }

```

```

1 >> ./a.out
2 9 8 7 6 5 4 3 2 1 0

```

queue

- Gagnagrindin `queue` geymir gögn og leyfir aðgang að fyrsta stakinu sem var bætt við.

Aðgerð	Tímaflækja
Bæta við staki	$\mathcal{O}(1)$
Lesa elsta stakið	$\mathcal{O}(1)$
Fjarlægja elsta stakið	$\mathcal{O}(1)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     int i, n = 10;
7     queue<int> a;
8     for (i = 0; i < 10; i++) a.push(i);
9     while (a.size() > 0)
10    {
11        printf("%d ", a.front());
12        a.pop();
13    }
14    printf("\n");
15    return 0;
16 }

```

```

1 >> ./a.out
2 0 1 2 3 4 5 6 7 8 9

```

- Gagnagrindin `set` geymir gögn án endurtekninga og leyfir hraða uppflettingu.

Aðgerð	Tímaflækja
Bæta við staki	$\mathcal{O}(\log n)$
Fjarlægja stak	$\mathcal{O}(\log n)$
Gá hvort staki hafi verið bætt við	$\mathcal{O}(\log n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     srand(time(NULL));
7     int i, n = 10;
8     set<int> a;
9     while (a.size() < 10) a.insert(rand()%100);
10    while (a.size() > 0)
11    {
12        printf("%2d ", *a.begin());
13        a.erase(a.begin());
14    }
15    printf("\n");
16    return 0;
17 }

```

```

1 >> ./a.out
2 2 9 18 29 34 42 43 46 91 97

```

