Lausnar aðferðir

Tæmandi leit og ráðug reiknirit

Bergur Snorrason

20. janúar 2020

Efnisyfirlit

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),
  - Gráðug reiknirit (e. *greedy algorithms*),

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),
  - Gráðug reiknirit (e. *greedy algorithms*),
  - Deila og drottna reiknirit (e. *divide and conquer algorithms*),

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),
  - Gráðug reiknirit (e. *greedy algorithms*),
  - Deila og drottna reiknirit (e. *divide and conquer algorithms*),
  - Kvik bestun (e. *dynamic programming*).

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),
  - Gráðug reiknirit (e. *greedy algorithms*),
  - Deila og drottna reiknirit (e. *divide and conquer algorithms*),
  - Kvik bestun (e. *dynamic programming*).
- Í dag verður farið í fyrstu tvær aðferðirnar.

Almennar nálganir lausna

- Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
  - Tæmandi leit eða ofbeldis aðferðin (e. *complete search, brute force*),
  - Gráðug reiknirit (e. *greedy algorithms*),
  - Deila og drottna reiknirit (e. *divide and conquer algorithms*),
  - Kvik bestun (e. *dynamic programming*).
- Í dag verður farið í fyrstu tvær aðferðirnar.
- Við styttum oft „tæmandi leit" í „CS" (fyrir „Complete search") og „kvik bestun" í „DP".

Efnisyfirlit

Allshersjar leit

- Þegar við erum að ræða dæmi kallast safn allra mögulega lausna *lausnarrúm* dæmisins.

Allshersjar leit

- Þegar við erum að ræða dæmi kallast safn allra mögulega lausna *lausnarrúm* dæmisins.
- *Tæmandi leit* felur í sér að leita í gegnum allt lausnarrúmið.

Allshersjar leit

- Þegar við erum að ræða dæmi kallast safn allra mögulega lausna *lausnarrúm* dæmisins.
- *Tæmandi leit* felur í sér að leita í gegnum allt lausnarrúmið.
- Tökum dæmi.

Allshersjar leit, dæmi

- Gefnar eru n mismunandi heiltölur. Hver þeirra er stærst?

Allshersjar leit, dæmi

- Gefnar eru n mismunandi heiltölur. Hver þeirra er stærst?
- Hér er lausnarrúmið einfaldlega þær tölur sem eru gefnar.

Allshersjar leit, dæmi

- Gefnar eru n mismunandi heiltölur. Hver þeirra er stærst?
- Hér er lausnarrúmið einfaldlega þær tölur sem eru gefnar.
- Okkur nægir að ítra í gegnum allar tölurnar og halda utan um stærstu töluna sem við höfum séð hingað til.

Allshersjar leit, dæmi

- Gefnar eru n mismunandi heiltölur. Hver þeirra er stærst?
- Hér er lausnarrúmið einfaldlega þær tölur sem eru gefnar.
- Okkur nægir að ítra í gegnum allar tölurnar og halda utan um stærstu töluna sem við höfum séð hingað til.
- Þessi aðferð er $\mathcal{O}(n)$.

Allshersjar leit, dæmi

- Gefnar eru n mismunandi heiltölur. Hver þeirra er stærst?
- Hér er lausnarrúmið einfaldlega þær tölur sem eru gefnar.
- Okkur nægir að ítra í gegnum allar tölurnar og halda utan um stærstu töluna sem við höfum séð hingað til.
- Þessi aðferð er $\mathcal{O}(n)$.
- Almennt fáum við að ef lausnarrúmið er af stærð m og við getum athugað hverja lasun í $\mathcal{O}(k)$, þá er CS $\mathcal{O}(mk)$.

Allshersjar leit, öll hlutmengi

- Dæmið hér á undan gæti leitt ykkur til að halda að CS sé alltaf einfalt. Það þarf þó ekki að vera.

Allshersjar leit, öll hlutmengi

- Dæmið hér á undan gæti leitt ykkur til að halda að CS sé alltaf einfalt. Það þarf þó ekki að vera.
- Tökum annað dæmi.

Allshersjar leit, öll hlutmengi

- Dæmið hér á undan gæti leitt ykkur til að halda að CS sé alltaf einfalt. Það þarf þó ekki að vera.
- Tökum annað dæmi.
- Gefin er runa af n tölum. Hver er lengsta vaxandi hlutruna gefnu rununnar?

Allshersjar leit, öll hlutmengi

- Dæmið hér á undan gæti leitt ykkur til að halda að CS sé alltaf einfalt. Það þarf þó ekki að vera.
- Tökum annað dæmi.
- Gefin er runa af n tölum. Hver er lengsta vaxandi hlutruna gefnu rununnar?
- Ef við viljum leysa þetta dæmi með CS þá þurfum við að skoða sérhvert hlutmengi gefnu rununnar.

Útúrdúr um hlutmengi

- Ef við erum með endanlegt mengi A af stærð n getum við númerað öll stökin með tölunum $1, 2, ..., n$.

Útúrdúr um hlutmengi

- Ef við erum með endanlegt mengi A af stærð n getum við númerað öll stökin með tölunum $1, 2, ..., n$.
- Sérhvert hlutmengi einkennist af því hvort stak k sé í hlutmenginu eða ekki, fyrir öll k í $1, 2, ..., n$.

Útúrdúr um hlutmengi

- Ef við erum með endanlegt mengi A af stærð n getum við númerað öll stökin með tölunum $1, 2, ..., n$.
- Sérhvert hlutmengi einkennist af því hvort stak k sé í hlutmenginu eða ekki, fyrir öll k í $1, 2, ..., n$.
- Við fáum þá að fjöldi hlutmengja í A er 2^n.

Útúrdúr um bitaframsetingu talna

- Fyrir hlutmengi H í A er til ótvíræð ákvörðuð tala b sem 1 í k-ta sæti bitaframsetningar sinnar þ.þ.a.a. k-ta stak A sé í H.

Útúrdúr um bitaframsetingu talna

- Fyrir hlutmengi H í A er til ótvírætt ákvörðuð tala b sem 1 í k-ta sæti bitaframsetningar sinnar þ.þ.a.a. k-ta stak A sé í H.
- Þetta gefur okkur gagntæka samsvörun milli hlutmengja A og talnanna $0, 1, ..., 2^n - 1$.

Útúrdúr um bitaframsetingu talna

- Fyrir hlutmengi H í A er til ótvírætt ákvörðuð tala b sem 1 í k-ta sæti bitaframsetningar sinnar þ.þ.a.a. k-ta stak A sé í H.
- Þetta gefur okkur gagntæka samsvörun milli hlutmengja A og talnanna $0, 1, ..., 2^n - 1$.
- Talan b er vanalega kölluð *bitakennir* eða *kennir* (e. *bitmask*, *mask*) hlutmengisins H.

Útúrdúr um bitaframsetingu talna

- Fyrir hlutmengi H í A er til ótvírætt ákvörðuð tala b sem 1 í k-ta sæti bitaframsetningar sinnar þ.þ.a.a. k-ta stak A sé í H.
- Þetta gefur okkur gagntæka samsvörun milli hlutmengja A og talnanna $0, 1, ..., 2^n - 1$.
- Talan b er vanalega kölluð *bitakennir* eða *kennir* (e. *bitmask*, *mask*) hlutmengisins H.
- Sem dæmi, ef $A = \{1, 2, 3, 4, 5, 6\}$ og $H = \{1, 3, 5, 6\}$ þá er $b = 110101_2 = 53$.

Útúrdúr um bitaframsetingu talna

- Fyrir hlutmengi H í A er til ótvírætt ákvörðuð tala b sem 1 í k-ta sæti bitaframsetningar sinnar þ.þ.a.a. k-ta stak A sé í H.
- Þetta gefur okkur gagntæka samsvörun milli hlutmengja A og talnanna $0, 1, ..., 2^n - 1$.
- Talan b er vanalega kölluð *bitakennir* eða *kennir* (e. *bitmask*, *mask*) hlutmengisins H.
- Sem dæmi, ef $A = \{1, 2, 3, 4, 5, 6\}$ og $H = \{1, 3, 5, 6\}$ þá er $b = 110101_2 = 53$.
- Kennir tómamengisins er alltaf 0 og kennir A er $2^n - 1$.

Útúrdúr um bitaframsetingu talna

- Þegar kemur að því að not bitakenni í forritun notum við okkur eftirfarandi:

Útúrdúr um bitaframsetingu talna

- Þegar kemur að því að not bitakenni í forritun notum við okkur eftirfarandi:

Kennir n-ta einstökungs	`1 << n`
Kennir fyllimengis kennis	`~A`
Kennir samengis tveggja kenna	`A|B`
Kennir sniðmengis tveggja kenna	`A&B`
Kennir samhverfs mismunar tveggja kenna	`A^B`
Kennir mismunar tveggja kenna	`A&(~B)`

Útúrdúr um bitaframsetingu talna

- Þegar kemur að því að not bitakenni í forritun notum við okkur eftirfarandi:

Kennir n-ta einstökungs	`1 << n`
Kennir fyllimengis kennis	`~A`
Kennir samengis tveggja kenna	`A|B`
Kennir sniðmengis tveggja kenna	`A&B`
Kennir samhverfs mismunar tveggja kenna	`A^B`
Kennir mismunar tveggja kenna	`A&(~B)`

- NB: Vegna forgang aðgerða í flestum forritunarmálum er góður vani að nota nóg af svigum þegar unnið er með bitaaðgerðir.

Útúrdúr um bitaframsetingu talna

- Þegar kemur að því að not bitakenni í forritun notum við okkur eftirfarandi:

Kennir n-ta einstökungs	`1 << n`
Kennir fyllimengis kennis	`~A`
Kennir samengis tveggja kenna	`A|B`
Kennir sniðmengis tveggja kenna	`A&B`
Kennir samhverfs mismunar tveggja kenna	`A^B`
Kennir mismunar tveggja kenna	`A&(~B)`

- NB: Vegna forgang aðgerða í flestum forritunarmálum er góður vani að nota nóg af svigum þegar unnið er með bitaaðgerðir.
- Til dæmis er `A&B == 0` jafngild `A&(B == 0)` í C++.

Allshersjar leit, öll hlutmengi

```c
#include <stdio.h>

int main()
{
    int n, i, j;
    scanf("%d", &n);
    int a[n];
    for (i = 0; i < n; i++) scanf("%d", &(a[i]));

    int mx = 0, mask;
    for (i = 0; i < (1 << n); i++)
    {
        int s[n], c = 0;
        for (j = 0; j < n; j++) if (((1 << j)&i) != 0) s[c++] = j;
        for (j = 1; j < c; j++) if (a[s[j]] < a[s[j - 1]]) break;
        if (j == c && c > mx) mx = c, mask = i;
    }

    printf("%d\n", mx);
    for (i = 0; i < n; i++) if (((1 << i)&mask) != 0) printf("%d ", a[i]);

    printf("\n");
    return 0;
}
```

Allshersjar leit, öll hlutmengi

- Hér er lausnarrúmið af stærð 2^n og við erum $\mathcal{O}(n)$ að ganga út skugga um hvort tiltekin lausn sé í raun rétt, svo reiknitíð er $\mathcal{O}(n2^n)$.

Allshersjar leit, allar umraðanir

- Við höfum oft áhuga á að ítra í gegnum allar umraðanir á lista talna.

Allshersjar leit, allar umraðanir

- Við höfum oft áhuga á að ítra í gegnum allar umraðanir á lista talna.
- Munum að, ef við höfum n ólíkar tölur þá getum við raðað þeim á $n! = 1 \cdot 2 \cdot ... \cdot n$ vegu.

Allshersjar leit, allar umraðanir

- Við höfum oft áhuga á að ítra í gegnum allar umraðanir á lista talna.
- Munum að, ef við höfum n ólíkar tölur þá getum við raðað þeim á $n! = 1 \cdot 2 \cdot ... \cdot n$ vegu.
- Tökum mjög einfalt dæmi:

Allshersjar leit, allar umraðanir

- Við höfum oft áhuga á að ítra í gegnum allar umraðanir á lista talna.
- Munum að, ef við höfum n ólíkar tölur þá getum við raðað þeim á $n! = 1 \cdot 2 \cdot ... \cdot n$ vegu.
- Tökum mjög einfalt dæmi:
- Gefið er n. Prentið allar umraðanir á $1, 2, ..., n$ í vaxandi stafrófsröð, hver á sinni línu.

Allshersjar leit, allar umraðanir

- Við höfum oft áhuga á að ítra í gegnum allar umraðanir á lista talna.
- Munum að, ef við höfum n ólíkar tölur þá getum við raðað þeim á $n! = 1 \cdot 2 \cdot ... \cdot n$ vegu.
- Tökum mjög einfalt dæmi:
- Gefið er n. Prentið allar umraðanir á $1, 2, ..., n$ í vaxandi stafrófsröð, hver á sinni línu.
- Við getum notað okkur innbyggða fallið `next_permutation` í C++.

Allshersjar leit, allar umraðanir

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n, i;
    cin >> n;
    vector<int> p;
    for (i = 0; i < n; i++) p.push_back(i + 1);
    do {
        for (i = 0; i < n; i++) cout << p[i] << ' ';
        cout << '\n';
    } while (next_permutation(p.begin(), p.end()));
    return 0;
}
```

Allshersjar leit, allar umraðanir endurkvæmt

```c
#include <stdio.h>
void perm(int* a, int* f, int n, int x)
{
    int i;
    if (x == n)
    {
        for (i = 0; i < n; i++) printf("%d ", a[i]);
        printf("\n");
        return;
    }
    for (i = 0; i < n; i++) if (f[i] == 0)
    {
        f[i] = 1;
        a[x] = i + 1;
        perm(a, f, n, x + 1);
        f[i] = 0;
    }
}

int main()
{
    int i, n;
    scanf("%d", &n);
    int a[n], f[n];
    for (i = 0; i < n; i++) f[i] = 0;
    perm(a, f, n, 0);
}
```

Allshersjar leit, allar umraðanir

- Mikilvægt er að p sé vaxandi röð til að fyrra forritið virki rétt, því lykkjan hættir þegar það lendir á síðustu umröðunni, í stafrófsröð.

Allshersjar leit, allar umraðanir

- Mikilvægt er að p sé vaxandi röð til að fyrra forritið virki rétt, því lykkjan hættir þegar það lendir á síðustu umröðunni, í stafrófsröð.
- Bæði eru þessi forrit $\mathcal{O}(n \cdot n!)$

Allshersjar leit, kostir og gallar

- Það eru ýmsir kostir við CS, til að mynda er lausnin sem reikniritin skila alltaf rétt (við sjáum á eftir aðferðir þar sem það gildir ekki) og CS lausnir eiga það til að vera auðveldar í útfærslu (það mætti segja að þær séu „eftir uppskrift").

Allshersjar leit, kostir og gallar

- Það eru ýmsir kostir við CS, til að mynda er lausnin sem reikniritin skila alltaf rétt (við sjáum á eftir aðferðir þar sem það gildir ekki) og CS lausnir eiga það til að vera auðveldar í útfærslu (það mætti segja að þær séu „eftir uppskrift").
- Eitt þarf samt að hafa í huga. Á keppnum eru CS dæmi yfirleitt flokkuð léttu dæmin, ef það er slíkt dæmi í keppninni á annað borð.

Allshersjar leit, kostir og gallar

- Það eru ýmsir kostir við CS, til að mynda er lausnin sem reikniritin skila alltaf rétt (við sjáum á eftir aðferðir þar sem það gildir ekki) og CS lausnir eiga það til að vera auðveldar í útfærslu (það mætti segja að þær séu „eftir uppskrift").
- Eitt þarf samt að hafa í huga. Á keppnum eru CS dæmi yfirleitt flokkuð léttu dæmin, ef það er slíkt dæmi í keppninni á annað borð.
- Keppnir innihalda frekar dæmi þar sem CS er aðeins hluti af lausninni.

Efnisyfirlit

Gráðug reiknirit

- Reiknirit sem tekur í hverju skrefi ákvörðun byggða á því hvað lítur best út á þeim tíma punkti kallast *gráðugt*.

Gráðug reiknirit

- Reiknirit sem tekur í hverju skrefi ákvörðun byggða á því hvað lítur best út á þeim tíma punkti kallast *gráðugt*.
- Höfum í huga að það er alls ekki sjálfsagt að gráðugt reiknirit skili réttri lausn.

Gráðug reiknirit

- Reiknirit sem tekur í hverju skrefi ákvörðun byggða á því hvað lítur best út á þeim tíma punkti kallast *gráðugt*.
- Höfum í huga að það er alls ekki sjálfsagt að gráðugt reiknirit skili réttri lausn.
- Tökum dæmi.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Dæmi um lausn er:

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Dæmi um lausn er:
- Látum k tákna dýrasta klinkið sem er ekki dýrara en n. Gefum til baka k og endurtökum fyrir $n - k$ þangað til n er 0.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Dæmi um lausn er:
- Látum k tákna dýrasta klinkið sem er ekki dýrara en n. Gefum til baka k og endurtökum fyrir $n - k$ þangað til n er 0.
- Tökum til dæmis $n = 24$. Þá myndum við gefa $10, 10, 1, 1, 1, 1$ til baka.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Dæmi um lausn er:
- Látum k tákna dýrasta klinkið sem er ekki dýrara en n. Gefum til baka k og endurtökum fyrir $n - k$ þangað til n er 0.
- Tökum til dæmis $n = 24$. Þá myndum við gefa $10, 10, 1, 1, 1, 1$ til baka.
- Það vill svo skemmtilega til að þessi gráðuga lausn virkar fyrir öll n.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 5 krónum og 10 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Dæmi um lausn er:
- Látum k tákna dýrasta klinkið sem er ekki dýrara en n. Gefum til baka k og endurtökum fyrir $n - k$ þangað til n er 0.
- Tökum til dæmis $n = 24$. Þá myndum við gefa $10, 10, 1, 1, 1, 1$ til baka.
- Það vill svo skemmtilega til að þessi gráðuga lausn virkar fyrir öll n.
- En hvað ef við breytum aðeins dæminu?

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 8 krónum og 20 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 8 krónum og 20 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Tökum til dæmis $n = 24$. Þá myndum aðferðin í glærunni á undan gefa $20, 1, 1, 1, 1$ til baka.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 8 krónum og 20 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Tökum til dæmis $n = 24$. Þá myndum aðferðin í glærunni á undan gefa $20, 1, 1, 1, 1$ til baka.
- En þetta er ekki besta leiðin. Betra væri að gefa $8, 8, 8$.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 8 krónum og 20 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Tökum til dæmis $n = 24$. Þá myndum aðferðin í glærunni á undan gefa $20, 1, 1, 1, 1$ til baka.
- En þetta er ekki besta leiðin. Betra væri að gefa $8, 8, 8$.
- Svo gráðuga aðferðin virkar bara stundum.

Gráðug reiknirit, dæmi

- Þú vinnur í sjoppu of þarft að gefa n krónur í afgang. Í boði er ótakmarkað magn af 1 krónum, 8 krónum og 20 krónum. Hver er minnsti fjöldi af klinki sem þú getur gefið?
- Tökum til dæmis $n = 24$. Þá myndum aðferðin í glærunni á undan gefa $20, 1, 1, 1, 1$ til baka.
- En þetta er ekki besta leiðin. Betra væri að gefa $8, 8, 8$.
- Svo gráðuga aðferðin virkar bara stundum.
- Þetta dæmi er þekkt sem Skiptimynta dæmið (e. *The Coin Change Problem*) og það er ekki augljóst hvenær gráðuga lausnin virkar.

Gráðug reiknirit, annað (hefðbundara) dæmi

- Þú ert yfirmaður hjá leigubíla fyrirtæki. Í dag mættu n bílstjórar í vinnuna og það eru m leigubílar til að skipa bílstjóra á. Ekki eru þó allir bílstrjórar og leigubílar skapaðir jafnir. Bíll i er h_i hestöfl og bílstjóri j getur keyrt bíla sem eru g_j hestöfl eða minna. Hver er mesti fjöld bíla sem þú getur skipað á bílstjóra, án þess að bílstjóri fá bíl umfram sína getu?

Gráðug reiknirit, annað (hefðbundara) dæmi

- Þú ert yfirmaður hjá leigubíla fyrirtæki. Í dag mættu n bílstjórar í vinnuna og það eru m leigubílar til að skipa bílstjóra á. Ekki eru þó allir bílstrjórar og leigubílar skapaðir jafnir. Bíll i er h_i hestöfl og bílstjóri j getur keyrt bíla sem eru g_j hestöfl eða minna. Hver er mesti fjöld bíla sem þú getur skipað á bílstjóra, án þess að bílstjóri fá bíl umfram sína getu?
- Tökum eftir að við viljum að bíll sé keyrður af þeim bílstrjóra sem er óreyndastur (en þó nógu hæfur til að keyra hann).

Gráðug reiknirit, annað (hefðbundara) dæmi

- Þú ert yfirmaður hjá leigubíla fyrirtæki. Í dag mættu n bílstjórar í vinnuna og það eru m leigubílar til að skipa bílstjóra á. Ekki eru þó allir bílstrjórar og leigubílar skapaðir jafnir. Bíll i er h_i hestöfl og bílstjóri j getur keyrt bíla sem eru g_j hestöfl eða minna. Hver er mesti fjöld bíla sem þú getur skipað á bílstjóra, án þess að bílstjóri fá bíl umfram sína getu?
- Tökum eftir að við viljum að bíll sé keyrður af þeim bílstrjóra sem er óreyndastur (en þó nógu hæfur til að keyra hann).
- Byrjum á að raða bílunum og bílstjórunum í vaxandi röð.

Gráðug reiknirit, annað (hefðbundara) dæmi

- Þú ert yfirmaður hjá leigubíla fyrirtæki. Í dag mættu n bílstjórar í vinnuna og það eru m leigubílar til að skipa bílstjóra á. Ekki eru þó allir bílstrjórar og leigubílar skapaðir jafnir. Bíll i er h_i hestöfl og bílstjóri j getur keyrt bíla sem eru g_j hestöfl eða minna. Hver er mesti fjöld bíla sem þú getur skipað á bílstjóra, án þess að bílstjóri fá bíl umfram sína getu?
- Tökum eftir að við viljum að bíll sé keyrður af þeim bílstrjóra sem er óreyndastur (en þó nógu hæfur til að keyra hann).
- Byrjum á að raða bílunum og bílstjórunum í vaxandi röð.
- Við getum núna í línulegum tíma fundið þann bílstrjóra sem getur keyrt fyrsta bíl.

Gráðug reiknirit, annað (hefðbundara) dæmi

- Þú ert yfirmaður hjá leigubíla fyrirtæki. Í dag mættu n bílstjórar í vinnuna og það eru m leigubílar til að skipa bílstjóra á. Ekki eru þó allir bílstrjórar og leigubílar skapaðir jafnir. Bíll i er h_i hestöfl og bílstjóri j getur keyrt bíla sem eru g_j hestöfl eða minna. Hver er mesti fjöld bíla sem þú getur skipað á bílstjóra, án þess að bílstjóri fá bíl umfram sína getu?
- Tökum eftir að við viljum að bíll sé keyrður af þeim bílstrjóra sem er óreyndastur (en þó nógu hæfur til að keyra hann).
- Byrjum á að raða bílunum og bílstjórunum í vaxandi röð.
- Við getum núna í línulegum tíma fundið þann bílstrjóra sem getur keyrt fyrsta bíl.
- Til að finna bílstjóra fyrir næstu bíla leitum við aftur línulega, en höldum áfram þaðan sem við hættum áðan.

Gráðug reiknirit, annað (hefðbundara) dæmi

- Teljum hversu marga bíla við náum að úthluta og sú tala er lausnin.

Gráðug reiknirit, annað (hefðbundara) dæmi

- Teljum hversu marga bíla við náum að úthluta og sú tala er lausnin.
- Þessi lausn er $\mathcal{O}(n + m)$.

Gráðug reiknirit, annað (hefðbundnara dæmi)

Að neðan er útfærsla á lausninni á glærunum hér á undan. Fyrstu tvær tölurnar á inntakinu eru n og m. Næstu n tölur lýsa getu bílstjóranna. Síðustu m tölurnar eru hestöfl bílanna.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n, m, i, j, x, r;
    cin >> n >> m;
    vector<int> a(n), b(m);
    for (i = 0; i < n; i++) cin >> a[i];
    for (i = 0; i < m; i++) cin >> b[i];
    sort(a.begin(), a.end());
    sort(b.begin(), b.end());
    r = j = 0;
    for (i = 0; i < m; i++)
    {
        while (j < n && a[j] < b[i]) j++;
        if (j < n) r++, j++;
    }
    cout << r << endl;
    return 0;
}
```

Efnisyfirlit

Samantekt

- Tæmandi leit er oft auðvelta að bera kennsl á, en á það til að vera of hæg. Hér má búast við TLE, en eitthvað furðulegt hefur gerst ef maður fær WA.

Samantekt

- Tæmandi leit er oft auðvelta að bera kennsl á, en á það til að vera of hæg. Hér má búast við TLE, en eitthvað furðulegt hefur gerst ef maður fær WA.
- Það er oft létt að semja gráðug reiknirit, en það getur verið mikil vinna að sanna að lausnin sem maður fær sé alltaf rétt. Gráðug reiknirit eiga ekki að fá TLE, en algengt er að fá WA.