# Danmarks Tekniske Universitet

| | |
|---|---|
| Kursusnavn | Introduktion til programmering og databehandling |
| Kursusnummer | 02631, 02632, 02633, 02634, 02692 |
| Hjælpemidler | Alle hjælpemidler er tilladt |
| Varighed | 2 timer |
| Vægtning | Opgaverne vægtes ens |

## Indhold



## Aflevering

Du skal aflevere dine løsninger elektronisk:

1. Du kan uploade dine løsninger individuelt på CodeJudge

   https://dtu.codejudge.net/

   under *Assignments* og *Exam*. Når du afleverer en løsning på CodeJudge bliver det test-eksempel som fremgår i opgavebeskrivelsen kørt på din løsning. Hvis din løsning består denne ene test, vil den fremgå som *Submitted*. Det betyder at din løsning består på dette ene test-eksempel. Du kan uploade til CodeJudge så mange gange som du ønsker under eksamen.

2. Du skal uploade alle dine løsninger på Eksamen site. Hver assignment skal uploades som en separat .py fil med samme navn som funktionen i opgaven:

   (a) `fit_windows.py`
   (b) `count_novel.py`
   (c) `count_peaks.py`
   (d) `classify_flower.py`
   (e) `qualify.py`

   Filerne skal afleveres separat (*ikke* som en zip-fil) og skal have præcis disse filnavne.

Efter eksamen vil dine løsninger fra DTU Inside blive automatisk evalueret på CodeJudge på en række forskellige tests for at undersøge om de generelt fungerer korrekt. Bedømmelsen af din aflevering foretages udelukkende på baggrund af hvor mange af de automatiserede test der bestås.

- Du skal sikre dig at din kode følger specifikationerne nøje.
- Hver løsning må ikke indeholde nogen yderligere kode udover den specificerede funktion, dog kan `import`-sætninger inkluderes.
- Husk at du kan tjekke om dine løsninger følger specifikationerne ved at uploade dem til CodeJudge.
- Bemærk at alle vektorer og matricer anvendt som input eller output skal være numpy arrays.

## Assignment A Fit Windows

Vi er givet en bygning med rektangulære sider der skal fittes med så mange vinduer som muligt på enten alle fire sider eller to af siderne (ikke top eller bunden af bygningen). Bygningen er specificeret med længde, bredde og højde og vinduerne er specificeret med bredde og højde. Vinduerne kan ikke roteres. De kan fittes så tæt som muligt. Antallet af vinduer skal være et heltal, så den totale sum af vinduesdimensioner er mindre eller lig med bygningsdimensionerne.

En ekstra parameter bestemmer om vinduerne skal fittes ved de to "length" (længde) sider, eller de to "width" (bredde) sider, eller med værdien "all" på alle fire sider.

Det totale antal vinduer, $c$, for alle de fire sider er (når "side" parameteren er sat til "all"):

$$c = 2 \times c_{length} \times c_{height} + 2 \times c_{width} \times c_{height}.$$

### Problemdefinition

Skriv en funktion med navnet `fit_windows` der som input tager en 3-element vektor **b** med længde, bredde og højde for bygningen og en 2-element vektor **w** med bredde og højde for vinduerne. En "side" parameter med værdien enten "length", "width" eller "all" bestemmer hvilke sider af bygningen hvor vinduerne skal fittes. Funktionen skal returnere antallet af vinduer der kan fittes til bygningen.

### Løsningsskabelon

```
def fit_windows(b, w, side):
  #insert your code
  return c
```

**Input**

| | |
|---|---|
| `b` | 3-element vektor med decimaltal, specificerende længde, bredde og højde. |
| `w` | 2-element vektor med decimaltal der specificere bredde og højde af vinduerne. |
| `side` | Streng der er enten "length" (indikerende at der er vinduer på de to længde-sider), "width" (indikerende at der er vinduer på de to bredde-sider) og "all" (indikerende vinduer på alle fire sider. |

**Output**

| | |
|---|---|
| `c` | Heltal med antallet af vinduer der kan fittes. |

### Eksempel

Betragt bygningsvektoren $\mathbf{b} = [37, 20, 10]$, vinduesvektoren $\mathbf{w} = [1.1, 1.75]$ og "side" parameteren med værdien "length". Antallet af vinduer der kan fittes i højden er

$$c_{height} = 5 \qquad \text{as } 5 \times 1.75 = 8.75 \leq 10.$$

Vinduer der kan fittes i længde-siden:

$$c_{length} = 33 \qquad \text{as } 33 \times 1.1 = 36.3 \leq 37$$

Vinduer der kunne fittes i bredde-siden (men som vi ignorerer da "side" parameteren er sat til "length"):

$$c_{width} = 18 \qquad \text{as } 18 \times 1.1 = 19.8 \leq 20$$

Det totale antal vinduer er:

$$c = 2 \times c_{length} \times c_{height} = 2 \times 33 \times 5 = 330 \tag{1}$$

Således skal værdien 330 returneres.

A

## Assignment B Count novel words

Givet en tekst repræsenteret i en streng med ord adskilt af mellemrum, ønsker vi at returnere en vektor med antallet er nye ord set indtil videre i teksten for hvert ord.

### Problemdefinition

Skriv en funktion med navnet `count_novel` som tager en streng som input og som returnerer en vektor med antallet af nye ord set indtil videre i teksten for hvert ord.

### Løsningsskabelon

```
def count_novel(text):
  # insert your code
  return c
```

**Input**

| | |
|---|---|
| `text` | Streng med en tekst med et eller flere ord. Ordene består af små bogstaver fra a til z og der er adskilt med et mellemrumstegn. |

**Output**

| | |
|---|---|
| `c` | Vektor med samme længde som antallet af ord i strengen og med et heltal med antallet af nye ord set indtil videre for hvert element. |

### Eksempel

Betragt teksten

"the man and another man walked down the street"

De nye ord i rækkefølge er:

the, man, and, another, (already seen), walked, down, (already seen), street

Det første ord "the" er et nyt ord og tæller som en. Det andet ord er "man" og også et nyt ord, så der er to nye ord indtil videre. Når vi kommer til det femte ord, som er "man", er dette ord ikke nyt og således for teksten indtil det femte ord er der 4 nye ord. Det fulde resultat er:

$$c = [1, 2, 3, 4, 4, 5, 6, 6, 7] \tag{2}$$

B

## Assignment C Count peaks

Givet en matrix ønsker vi at tælle antallet af toppunkter, der er defineret som elementer hvor differencen mellem elementet og mindst et af dets naboelementer er større end eller lig med 2. Naboområdet for et matrixelement er defineret som de fire matrixelementer lige over, under, til venstre eller til højre (for elementer ved kanten er der kun 2 eller 3 elementer).

### Problemdefinition

Skriv en funktion med navnet `count_peaks` der tager en matrix som input og returnerer antallet af toppunkter hvor differencen mellem elementet og en eller flere af dets naboer er større end eller lig med 2.

### Løsningsskabelon

```
def count_peaks(A):
  # insert your code
  return c
```

**Input**

| | |
|---|---|
| `A` | Matrix med en eller flere kolonner og en eller flere rækker med decimaltal. |

**Output**

| | |
|---|---|
| `c` | Ikke-negativt heltal med antal. |

### Eksempel

Betragt matricen

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 5 & 3 & 7 & 7.1 \\ 3 & 3 & 3 & 4 & 4 & 7 & 7.3 \end{bmatrix}. \tag{3}$$

Her har elementet $a_{2,1} = 3$ en nabo, $a_{1,1}$, hvor difference er $\geq 2$: $a_{2,1} - a_{1,1} = 3 - 1 = 2$. Elementet $a_{1,4} = 5$ har to naboelementer hvor differencen er større end eller lig med 2, f.eks. $a_{1,4} - a_{1,3} = 3 - 1 = 2$. Ydermere har elementet $a_{1,6}$ en nabo der er tilstrækkeligt lille: $a_{1,6} - a_{1,5} = 7 - 3 = 4$. Til sidst har elementet $a_{2,6}$ også en lille nabo: $a_{2,6} - a_{2,5} = 7 - 4 = 3$.

Følgende elementer er toppunkter:

$$\text{peaks} = \{a_{2,1}, a_{1,4}, a_{1,6}, a_{2,6}\} \tag{4}$$

Det totale antal elementer i dette sæt er $\underline{4}$ som er værdien der returneres.

C

## Assignment D Classify flower

Vi ønsker at kategorisere blomster med hensyn til en database repræsenteret i en matrix $\mathbf{D}$ af størrelsen $(N \times F)$, hvor $N$ er antallet af forskellige blomsterarter og $F$ er antallet af kendetegn. Med undtagelse af den sidste kolonne i matricen er kendetegnene kodet som ikke-negative heltal. Matricens sidste kolonne er speciel og koder blomstens typiske højde som et decimaltal.

En "søge-blomst" er repræsenteret i en vektor med længde $F$ hvor de første $F-1$ elementer er enten ikke-negative heltal eller har værdien nul der indikerer ubestemte elementer (kendetegn). Det sidste element indeholder højden som et decimaltal.

Vi ønsker at vide hvilke rækker i databasen der matcher søge-vektoren, mens elementerne (kendetegnene) i søge-vektoren der er ubestemte og indikeret med nul skal ignoreres. Ydermere ønsker vi at sortere de matchende rækker (blomster) i databasen med hensyn til højde, så rækkeindeks for de arter der er tættest i højde er returneret som de første elementer.

### Problemdefinition

Skriv en funktion med navnet `classify_flower` der som input tager en matrix $\mathbf{D}$, en søge-vektor $\mathbf{q}$ og returnerer en vektor, $\mathbf{s}$, med rækkeindeks (indekseret fra et) for de arter repræsenteret i rækker i $\mathbf{D}$ der matcher $\mathbf{q}$, mens elementer i søge-vektoren der er nul bliver ignoreret. Række-indekset i $\mathbf{s}$ skal sorteres i stigende rækkefølge baseret på højde-afstanden mellem søge- og databaseblomsterne.

### Løsningsskabelon

```
def classify_flower(D, q):
    #insert your code
    return s
```

**Input**

| | |
|---|---|
| `D` | Matrix med størrelsen $(N \times F)$ hvor $N \geq 1$ og $F \geq 2$ |
| `q` | Vektor med længde $F$ der repræsentere en "søge-blomst" hvor nul indikerer ubestemt kendetegn. |

**Output**

| | |
|---|---|
| `s` | Vektor med række-indeks for de arter der matcher søge-blomster og sorteret med hensyn til hvor tæt de er i højde i forhold til søge-vektoren. Indekseret fra et. |

### Eksempel

Som et eksempel betragt en database der repræsenterer 4 blomster og deres 5 kendetegn:

$$\mathbf{D} = \begin{bmatrix} 4 & 1 & 2 & 3 & 10.1 \\ 4 & 1 & 2 & 2 & 20.5 \\ 4 & 2 & 3 & 3 & 25.7 \\ 5 & 2 & 3 & 1 & 19.4 \end{bmatrix} \tag{5}$$

For en eksempel-søge-vektor $\mathbf{q}$ er kendetegnene #2 og #3 ubestemte (og indikeret med et nul) og højden af blomsten er 21.1:

$$\mathbf{q} = [4, 0, 0, 3, 21.1] \tag{6}$$

Her matcher det første kendetegn ($q_1 = 4$) rækkerne (arterne) 1, 2 og 3, mens det fjerde kendetegn ($q_4 = 3$) matcher rækkerne 1 og 3, så kun række 1 og 3 matcher alle bestemte kendetegn. Når vi sammenligner højderne er differencerne:

$$\Delta_1 = |d_{1,5} - q_5| = |10.1 - 21.1| = 10, \qquad \text{and} \qquad \Delta_3 = |d_{3,5} - q_5| = |25.7 - 21.1| = 4.6. \tag{7}$$

Her er arterne i den tredje række tættest, så vi sorterer resultaterne som $\mathbf{s} = [3, 1]$, det vil sige vi returnere $\underline{[3, 1]}$.

D

## Assignment E Football qualification

Givet to matricer, $\mathbf{A}$ og $\mathbf{R}$, der koder holdfordelingen og antal mål i en gruppeturnering, ønsker vi at bestemme hvilke to hold der kvalificerer sig til avancement i turneringen. Som et eksempel betragt UEFA Euro 1992 gruppe 1 med Sverige, Frankrig, Danmark og England, der er kodet som henholdsvis 1, 2, 3 og 4. I den første kamp spillede Sverige mod Frankrig med resultatet 1 – 1. Den næste kamp var Danmark mod England med resultatet 0 – 0. Den fjerde kamp var Sverige mod Danmark med resultatet 1 – 0. Med det fulde sæt af resultater er $\mathbf{A}$ og $\mathbf{R}$ matricerne:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 2 & 4 \\ 1 & 3 \\ 1 & 4 \\ 2 & 3 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{8}$$

I hvert spil modtager det vindende hold to point og det tabende hold nul point. Uafgjort vil resultere i et point til hvert hold.

I det ovenstående eksempel har Sverige (1) vundet to kampe og har en uafgjort, resulterende i 5 point. Frankrig (2) har to uafgjort og et tab, resulterende i 2 point. Danmark (3) har en uafgjort og et tab og en vundet kamp, resulterende i 3 point, mens England har to uafgjort og et tab, resulterende i 2 point. Sverige og Danmark er de to tophold og kvalificerer sig med Sverige først.

### Problemdefinition

Skriv en funktion med navnet `qualify` som tager en matrix, $\mathbf{A}$ af størrelsen $(G \times 2)$, med holdfordelinger for $G$ kampe og en matrix, $\mathbf{R}(G \times 2)$, der indikerer antallet af mål. Funktionen skal returnere hold-identifikatorerne for de to hold der kvalificerer sig, og sorteret så at holdet med de fleste point er returneret først. Det kan antages at de vindende hold altid er adskilt i point (så det ikke er nødvendigt at kikke på måldifferencen som i virkelighedens verden). Holdene er kodet fra 1 og i trin af 1 og op til antallet af hold. Ikke alle holdkombinationer spilles nødvendigvis.

### Løsningsskabelon

```
def qualify(A, R):
  # insert your code
  return t
```

**Input**

| | |
|---|---|
| `A` | $(G \times 2)$-matrix hvor hver række indeholder identifikatorer for holdene der spiller i en specifik kamp, hvor $G \geq 1$. |
| `R` | $(G \times 2)$-matrix med mål scoret i hver kamp, hvor $G \geq 1$. |

**Output**

| | |
|---|---|
| `t` | 2-element vektor med identifikatorer for hold der kvalificerer sig til at avancere i turneringen. |

### Eksempel

I det overstående UEFA Euro 1992 gruppe 1 eksempel skal identifikatorerne for Sverige og Danmark returneres med Sverige først i vektoren, $t = \underline{[1, 3]}$.

E