Köer

Programmering 2

vt 24

Outline

Kö

En *kö* är en datastruktur där varje element refererar till ett annat element, som en i stack. Skillnaden mellan en kö och en stack är att i en kö så är det ”först in först ut” (FIFO) som gäller.

Exempel

En vanlig *kö* IRL är kön till kassan i en affär. Där ställer man sig på led och den som kom först blir betjänad först.

Kö

En *kö* har följande fyra metoder:

1. put (enqueue)
2. get (dequeue)
3. front
4. rear

Metoder

- Metoden `put` lägger till ett element sist i *kön*. Du kan jämföra det med metoden `append` i listor.

Metoder

- Metoden `put` lägger till ett element sist i *kön*. Du kan jämföra det med metoden `append` i listor.
- Metoden `get` plockar bort det första elementet i *kön*. Du kan jämföra det med motsvarande metod i listor.

Metoder

- Metoden `put` lägger till ett element sist i *kön*. Du kan jämföra det med metoden `append` i listor.
- Metoden `get` plockar bort det första elementet i *kön*. Du kan jämföra det med motsvarande metod i listor.
- Metoden `front` tittar på det första elementet i *kön* och ger värdet.

Metoder

- Metoden `put` lägger till ett element sist i *kön*. Du kan jämföra det med metoden `append` i listor.
- Metoden `get` plockar bort det första elementet i *kön*. Du kan jämföra det med motsvarande metod i listor.
- Metoden `front` tittar på det första elementet i *kön* och ger värdet.
- Metoden `rear` tittar på det sista elementet i *kön* och ger värdet.
  En kö måste inte ha metoden `rear`

Outline

Listor som köer

Listor

I Python kan du simulera en *kö* med en lista. Det är ganska enkelt eftersom listan har alla metoder en *kö* har och lite till.

```
1 kö = []
2 kö.append(4) # Motsvarar put
3 kö.pop(0) # Motsvarar get
4 kö[0] # Motsvarar front
5 kö[-1] # Motsvarar rear
```

Köer och listor

Nackdelen med listor

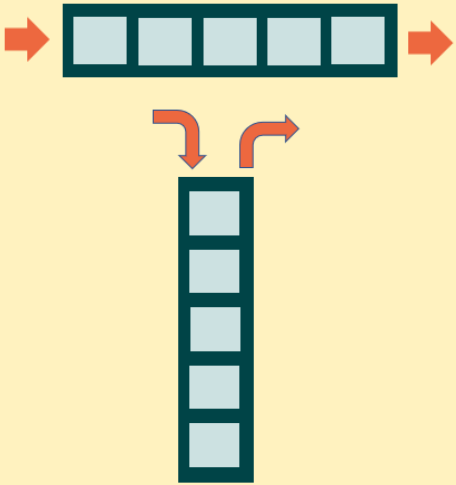
Nackdelen med listor är att listor är en sammanhängande grupp i minnet på datorn. Så om man lägger till något i en lista så tar det mer plats i minnet – vilket är som det ska – men om det nu skulle krocka med en annan sak lagrat i minnet så allokerar man en ny plats i minnet och flyttar hela listan i minnet. Detta gör att operationer på listor kan ta väldigt olika lång tid, om datorn måste flytta på hela listan i minnet.

Köer och listor

Fördelen med köer

Köer däremot lagrar varje element separat i minnet, det kan ge lite mer overhead i varje element. Men det gör att man riskerar inte att flytta på hela *kön* när man lägger till element i den. Detta gör att operationer med *köer* är konsekventa i hur lång tid de tar att genomföra.

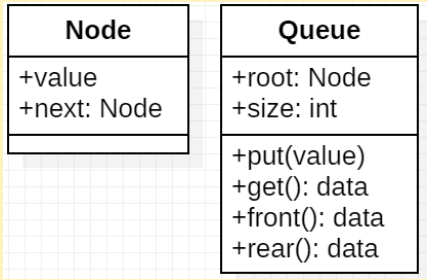
Kö vs. stack



Outline

Övningar

Skapa din egna `queue`-klass. Du får en driver kod på Vklass döpt till `kö.py`
Här ser du klassdiagramen för klasserna `Node` och `Queue`



En del implementationer har istället för attributet `root` attributen `first` och `last`. Genom att använda `last` kan man korta ner tiden det tar att lägga till ett nytt element.