# Page Replacement Algorithm

By
Prutchakorn 5912046

# Page Fault

Process <--> Memory

When the process access a page that is not in the memeory

# Page Replacement Algorithm

When page fault occurs system needs to find free frame to swap

When there is no free frame left, then the system will use page replacement algorithm

# 3 Page Replacement Algorithm

- Optimal
- LRU (Least Recently Used)
- FIFO

# Optimal

```
Number = [int(x) for x in input('Number: ')]
n = len(Number)
m = int(input("Number of Frame: "))
x = 0
page_faults = 0
page = []
FREE = -1
for i in range(m):
    page.append(FREE)

for i in range(n):
    flag = 0
    for j in range(m):
        if page[j] == Number[i]:
            flag = 1
            break

    if flag == 0:
        # look for an empty one
        faulted = False
        new_slot = FREE
        for q in range(m):
            if page[q] == FREE:
                faulted = True
                new_slot = q

        if not faulted:
            # find next use farthest in future
            max_future = 0
            max_future_q = FREE
            for q in range(m):
                if page[q] != FREE:
                    found = False
                    for ii in range(i, n):
                        if Number[ii] == page[q]:
                            found = True
                            if ii > max_future:
                                # print "\n\tFound what will be used last: a[%d] = %d" % (ii, a[ii]),
                                max_future = ii
                                max_future_q = q

                            break

                    if not found:
                        # print "\n\t%d isn't used again." % (page[q]),
                        max_future_q = q
                        break
```

```
            faulted = True
            new_slot = max_future_q

        page_faults += 1
        page[new_slot] = Number[i]
        print("\n%d ->" % (Number[i]))
        for j in range(m):
            if page[j] != FREE:
                print(page[j])
            else:
                print("-")
    else:
        print("\n%d -> No Page Fault" % (Number[i]))

    print("\n Total page faults : %d." % (page_faults - 3))
```

# LRU

```
Number = [int(x) for x in input('Number: ')]
n = len(Number)
m = int(input("Number of Frame: "))
x = 0
page_faults = 0
page = []
for i in range(m):
    page.append(-1)

for i in range(n):
    flag = 0
    for j in range(m):
        if page[j] == Number[i]:
            flag = 1
            break

    if flag == 0:
        if page[x] != -1:
            min = 999
            for k in range(m):
                flag = 0
                j = i
                while j >= 0:
                    j -= 1
                    if page[k] == Number[j]:
                        flag = 1
                        break
                if flag == 1 and min > j:
                    min = j
                    x = k

        page[x] = Number[i]
        x = (x + 1) % m
        page_faults += 1
        print("\n%d ->" % (Number[i]))
        for j in range(m):
            if page[j] != -1:
                print(page[j])
            else:
                print("-")
    else:
        print("\n%d -> No Page Fault" % (Number[i]))

    print("\n Total page faults : %d." % (page_faults - 3))
```

# FIFO

```
Number = [int(x) for x in input('Number: ')]
n = len(Number)
m = int(input("Number of Frame: "))
f = -1
page_faults = 0
page = []
for i in range(m):
    page.append(-1)

for i in range(n):
    flag = 0
    for j in range(m):
        if page[j] == Number[i]:
            flag = 1
            break

    if flag == 0:
        f = (f + 1) % m
        page[f] = Number[i]
        page_faults += 1
        print("\n%d ->" % (Number[i]))
        for j in range(m):
            if page[j] != -1:
                print(page[j])
            else:
                print("-")
    else:
        print("\n%d -> No Page Fault" % (Number[i]))

    print("\n Total page faults : %d." % (page_faults - 3))
```