



BAB	: ALOKASI MEMORI
NAMA	: DANI ADRIAN
NIM	: 225150201111009
TANGGAL	: 02/05/2023
ASISTEN	: ZHAFRAN RAMA AZMI
	GIBRAN HAKIM

7.4. Langkah Praktikum

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1. Lakukan percobaan pada kode program berikut.

7.4.1. Solusi first-fit

```
1. #include <stdio.h>
2. // mendefinisikan nilai variabel array
   maksimum sejumlah 20
3. #define max 20
4. void main()
5. {
6.     // set setiap variabel yang dibutuhkan
7.     int frag[max], ukuranBlok[max],
   ukuranProses[max], i, j, jmlBlok, jmlProses,
   temp;
8.     static int alokasiBlok[max], flags[max];
9.
10.    // input dari pengguna
11.    printf("\nTeknik Alokasi Memori - First
   Fit");
12.    printf("\nMasukkan jumlah blok memori:");
13.    scanf("%d", &jmlBlok);
14.    printf("Masukkan jumlah proses:");
15.    scanf("%d", &jmlProses);
16.
17.    printf("\nMasukkan ukuran blok memori:-\n");
18.    for (i = 1; i <= jmlBlok; i++)
19.    {
20.        printf("Blok %d:", i);
21.        scanf("%d", &ukuranBlok[i]);
22.    }
23.
24.    printf("Masukkan ukuran proses :-\n");
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
25.     for (i = 1; i <= jmlProses; i++)
26.     {
27.         printf("Proses %d:", i);
28.         scanf("%d", &ukuranProses[i]);
29.     }
30.
31.     // mengambil setiap proses satu persatu
32.     for (i = 1; i <= jmlProses; i++)
33.     {
34.         for (j = 1; j <= jmlBlok; j++)
35.         {
36.
37.             // melakukan proses alokasi dengan
mengurangi ukuran blok dengan ukuran proses
38.             if (alokasiBlok[j] != 1)
39.             {
40.                 temp = ukuranBlok[j] -
ukuranProses[i];
41.                 if (temp >= 0)
42.                 {
43.                     flags[i] = j;
44.                     break;
45.                 }
46.                 else
47.                 {
48.                     temp = 0;
49.                 }
50.             }
51.         }
52.
53.         // alokasi first-fit berdasarkan informasi
alokasi blok memori
54.         frag[i] = temp;
55.         alokasiBlok[flags[i]] = 1;
56.     }
57.
58.     // tampilan output
59.     printf("\nNo. Proses:\tUkuran Proses
:\tAlokasi Blok:\tUkuran Blok:\tFragment");
60.     for (i = 1; i <= jmlProses; i++)
61.         printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i,
ukuranProses[i], flags[i],
ukuranBlok[flags[i]], frag[i]);
62. }
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

- a. Simpan berkas tersebut dengan nama alloc-first-fit.c

```
GNU nano 6.2                                     New Buffer *
printf("Blok %d:", i);
scanf("%d", &ukuranBlok[i]);
}

printf("Masukkan ukuran proses :-\n");
for (i = 1; i <= jmlProses; i++)
{
    printf("Proses %d:", i);
    scanf("%d", &ukuranProses[i]);
}

// mengambil setiap proses satu persatu
for (i = 1; i <= jmlProses; i++)
{
    for (j = 1; j <= jmlBlok; j++)
    {
        // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
        if (alokasiBlok[j] != 1)
        {
            temp = ukuranBlok[j] - ukuranProses[i];
            if (temp >= 0)
            {
                flags[i] = j;
                break;
            }
            else
            {
                temp = 0;
            }
        }
    }

    // alokasi first-fit berdasarkan informasi alokasi blok memori
    frag[i] = temp;
    alokasiBlok[flags[i]] = 1;
}

// tampilan output
printf("\nNo. Proses:\tUkuran Proses :\tAlokasi Blok:\tUkuran Blok:\tFragment");
for (i = 1; i <= jmlProses; i++)
    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, ukuranProses[i], flags[i], ukuranBlok[flags[i]], frag[i]);
}

File Name to Write: alloc-first-fit.c
^G Help      M-D DOS Format      M-A Append
^C Cancel    M-M Mac Format      M-P Prepend
```

- b. Kompilasi program alloc-first-fit.c melalui terminal dengan menuliskan perintah

gcc alloc-first-fit.c -o alloc-first-fit

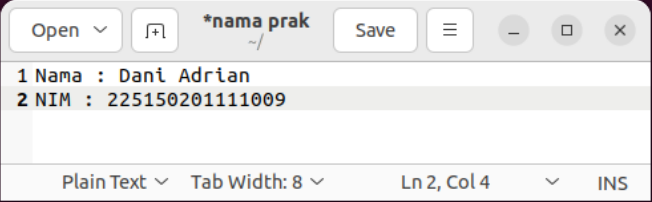
```
userlinux@Linux:~$ gcc alloc-first-fit.c -o alloc-first-fit
userlinux@Linux:~$
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

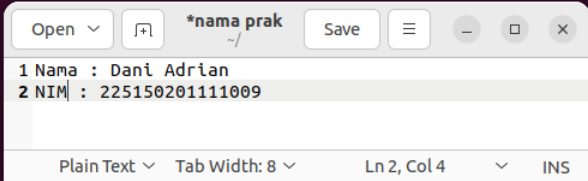
- c. Jalankan program dengan menuliskan perintah `./alloc-first-fit.exe`

```
userlinux@Linux:~$ ./alloc-first-fit
Teknik Alokasi Memori - First Fit
Masukkan jumlah blok memori:
```



- d. Masukkan variable berikut: Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

```
userlinux@Linux:~$ ./alloc-first-fit
Teknik Alokasi Memori - First Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4
Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14
```



- e. Capture outputnya dan simpan sebagai laporan.



```
userlinux@Linux:~$ ./alloc-first-fit
Teknik Alokasi Memori - First Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14

No. Proses:   Ukuran Proses :  Alokasi Blok:   Ukuran Blok:   Fragment
1             2                1             15             13
2             5                2             20             15
3             6                4              7              1
4            14                0              0              0
0userlinux@Linux:~$
```

f. Amati output hasil percobaannya untuk referensi pada pembahasan.

```
No. Proses:   Ukuran Proses :  Alokasi Blok:   Ukuran Blok:   Fragment
1             2                1             15             13
2             5                2             20             15
3             6                4              7              1
4            14                0              0              0
0userlinux@Linux:~$
```

7.4.2. Solusi best-fit

```
1. #include <stdio.h>
2. // mendefinisikan nilai variabel array
   maksimum sejumlah 20
3. #define max 20
4. void main()
5. {
6.     // set setiap variabel yang dibutuhkan
7.     int frag[max], ukuranBlok[max],
   ukuranProses[max], i, j, jmlBlok, jmlProses,
   temp, lowest = 10000;
8.     static int alokasiBlok[max], flags[max];
9.
10.    // input dari pengguna
11.    printf("\nTeknik Alokasi Memori - Best
   Fit");
12.    printf("\nMasukkan jumlah blok memori:");
13.    scanf("%d", &jmlBlok);
14.    printf("Masukkan jumlah proses:");
15.    scanf("%d", &jmlProses);
16.
17.    printf("\nMasukkan ukuran blok memori:-\n");
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
18.   for (i = 1; i <= jmlBlok; i++)
19.   {
20.       printf("Blok %d:", i);
21.       scanf("%d", &ukuranBlok[i]);
22.   }
23.
24.   printf("Masukkan ukuran proses :-\n");
25.   for (i = 1; i <= jmlProses; i++)
26.   {
27.       printf("Proses %d:", i);
28.       scanf("%d", &ukuranProses[i]);
29.   }
30.
31.   for (i = 1; i <= jmlProses; i++)
32.   {
33.       for (j = 1; j <= jmlBlok; j++)
34.       {
35.
36.           // melakukan proses alokasi dengan
mengurangi ukuran blok dengan ukuran proses
37.           if (alokasiBlok[j] != 1)
38.           {
39.               temp = ukuranBlok[j] -
ukuranProses[i];
40.               if (temp >= 0)
41.                   if (lowest > temp)
42.                   {
43.                       flags[i] = j;
44.                       lowest = temp;
45.                   }
46.           }
47.       }
48.
49.       // alokasi best-fit berdasarkan nilai
fragment terkecil
50.       frag[i] = lowest;
51.       alokasiBlok[flags[i]] = 1;
52.       lowest = 10000;
53.   }
54.
55.   // tampilan output
56.   printf("\nNo. Proses:\tUkuran Proses
:\tAlokasi Blok:\tUkuran Blok:\tFragment");
57.   for (i = 1; i <= jmlProses; i++)
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
58.     printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i,
        ukuranProses[i], flags[i],
        ukuranBlok[flags[i]], frag[i]);
59. }
```

a. Simpan berkas tersebut dengan nama alloc-best-fit.c

```
GNU nano 6.2                                     New Buffer *
#include <stdio.h>
// mendefinisikan nilai variabel array maksimum sejumlah 20
#define max 20
void main()
{
    // set setiap variabel yang dibutuhkan
    int frag[max], ukuranBlok[max], ukuranProses[max], i, j, jmlBlok, jmlProses, temp, lowest = 10000;
    static int alokasiBlok[max], flags[max];

    // input dari pengguna
    printf("\nTeknik Alokasi Memori - Best Fit");
    printf("\nMasukkan jumlah blok memori:");
    scanf("%d", &jmlBlok);
    printf("Masukkan jumlah proses:");
    scanf("%d", &jmlProses);

    printf("\nMasukkan ukuran blok memori:-\n");
    for (i = 1; i <= jmlBlok; i++)
    {
        printf("Blok %d:", i);
        scanf("%d", &ukuranBlok[i]);
    }

    printf("Masukkan ukuran proses :-\n");
    for (i = 1; i <= jmlProses; i++)
    {
        printf("Proses %d:", i);
        scanf("%d", &ukuranProses[i]);
    }

    for (i = 1; i <= jmlProses; i++)
    {
        for (j = 1; j <= jmlBlok; j++)
        {
            // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
            if (alokasiBlok[j] != 1)
            {
                temp = ukuranBlok[j] - ukuranProses[i];
                if (temp >= 0)
                {
                    if (lowest > temp)
                    {
                        flags[i] = j;
                        lowest = temp;
                    }
                }
            }
        }
    }
}
```

b. Kompilasi program alloc-first-fit.c melalui terminal dengan menuliskan perintah

gcc alloc-best-fit.c -o alloc-best-fit

```
userlinux@Linux:~$ gcc alloc-best-fit.c -o alloc-best-fit
userlinux@Linux:~$
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

- c. Jalankan program dengan menuliskan perintah **./alloc-best-fit.exe**

```
userlinux@linux:~$ ./alloc-best-fit
Teknik Alokasi Memori - Best Fit
Masukkan jumlah blok memori:
Masukkan jumlah proses:
```

Open

*nama prak

Save

1 Nama : Dani Adrian

2 NIM : 225150201111009

- d. Masukkan variable berikut: Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

```
userlinux@Linux:~$ ./alloc-best-fit
Teknik Alokasi Memori - Best Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4
Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14
```

Open

*nama prak

Save

1 Nama : Dani Adrian

2 NIM : 225150201111009

Plain Text

Tab Width: 8

Ln 2, Col 4

INS

- e. Capture outputnya dan simpan sebagai laporan.

```
userlinux@Linux:~$ ./alloc-best-fit
Teknik Alokasi Memori - Best Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4
Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14
```

No. Proses:	Ukuran Proses :	Alokasi Blok:	Ukuran Blok:	Fragment
1	2	5	4	2
2	5	3	5	0
3	6	4	7	1
4	14	1	15	1

```
userlinux@Linux:~$
```

Open

*nama prak

Save

1 Nama : Dani Adrian

2 NIM : 225150201111009

Plain Text

Tab Width: 8

Ln 2, Col 4

INS

- f. Amati output hasil percobaannya untuk referensi pada pembahasan.

No. Proses:	Ukuran Proses :	Alokasi Blok:	Ukuran Blok:	Fragment
1	2	5	4	2
2	5	3	5	0
3	6	4	7	1
4	14	1	15	1

```
userlinux@Linux:~$
```

Open

*nama prak

Save

1 Nama : Dani Adrian

2 NIM : 225150201111009

Plain Text

Tab Width: 8

Ln 2, Col 4

INS



7.4.3. Solusi worst-fit

```
1. #include <stdio.h>
2. // mendefinisikan nilai variabel array
   maksimum sejumlah 20
3. #define max 20
4. void main()
5. {
6.     // set setiap variabel yang dibutuhkan
7.     int frag[max], ukuranBlok[max],
   ukuranProses[max], i, j, jmlBlok, jmlProses,
   temp, highest = 0;
8.     static int alokasiBlok[max], flags[max];
9.
10.    // input dari pengguna
11.    printf("\nTeknik Alokasi Memori - Worst
   Fit");
12.    printf("\nMasukkan jumlah blok memori:");
13.    scanf("%d", &jmlBlok);
14.    printf("Masukkan jumlah proses:");
15.    scanf("%d", &jmlProses);
16.
17.    printf("\nMasukkan ukuran blok memori:-\n");
18.    for (i = 1; i <= jmlBlok; i++)
19.    {
20.        printf("Blok %d:", i);
21.        scanf("%d", &ukuranBlok[i]);
22.    }
23.
24.    printf("Masukkan ukuran proses :-\n");
25.    for (i = 1; i <= jmlProses; i++)
26.    {
27.        printf("Proses %d:", i);
28.        scanf("%d", &ukuranProses[i]);
29.    }
30.
31.    for (i = 1; i <= jmlProses; i++)
32.    {
33.        for (j = 1; j <= jmlBlok; j++)
34.        {
35.
36.            // melakukan proses alokasi dengan
   mengurangi ukuran blok dengan ukuran proses
37.            if (alokasiBlok[j] != 1)
38.            {
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
39.         temp = ukuranBlok[j] -
ukuranProses[i];
40.         if (temp >= 0)
41.             if (highest < temp)
42.                 {
43.                     flags[i] = j;
44.                     highest = temp;
45.                 }
46.             }
47.         }
48.
49.         // alokasi worst-fit berdasarkan nilai
fragment terbesar
50.         frag[i] = highest;
51.         alokasiBlok[flags[i]] = 1;
52.         highest = 0;
53.     }
54.
55.     // tampilan output
56.     printf("\nNo. Proses:\tUkuran Proses
:\tAlokasi Blok:\tUkuran Blok:\tFragment");
57.     for (i = 1; i <= jmlProses; i++)
58.         printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i,
ukuranProses[i], flags[i],
ukuranBlok[flags[i]], frag[i]);
59. }
```

- a. Simpan berkas tersebut dengan nama alloc-worst-fit.c



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
GNU nano 6.2                                     New Buffer *
#include <stdio.h>
// mendefinisikan nilai variabel array maksimum sejumlah 20
#define max 20
void main()
{
    // set setiap variabel yang dibutuhkan
    int frag[max], ukuranBlok[max], ukuranProses[max], i, j, jmlBlok, jmlProses, temp, highest = 0;
    static int alokasiBlok[max], flags[max];

    // input dari pengguna
    printf("\nTeknik Alokasi Memori - Worst Fit");
    printf("\nMasukkan jumlah blok memori:");
    scanf("%d", &jmlBlok);
    printf("Masukkan jumlah proses:");
    scanf("%d", &jmlProses);

    printf("\nMasukkan ukuran blok memori: -\n");
    for (i = 1; i <= jmlBlok; i++)
    {
        printf("Blok %d:", i);
        scanf("%d", &ukuranBlok[i]);
    }

    printf("Masukkan ukuran proses: -\n");
    for (i = 1; i <= jmlProses; i++)
    {
        printf("Proses %d:", i);
        scanf("%d", &ukuranProses[i]);
    }

    for (i = 1; i <= jmlProses; i++)
    {
        for (j = 1; j <= jmlBlok; j++)
        {
            // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
            if (alokasiBlok[j] != 1)
            {
                temp = ukuranBlok[j] - ukuranProses[i];
                if (temp >= 0)
                {
                    if (highest < temp)
                    {
                        flags[i] = j;
                        highest = temp;
                    }
                }
            }
        }
    }
}
```

Open

*nama prak

Save

Ln 2, Col 4

INS

1 Nama : Dani Adrian

2 NIM : 225150201111009

File Name to Write: alloc-worst-fit.c

^O Help

^C Cancel

M-D DOS Format

M-M Mac Format

M-A Append

M-P Prepend

- b. Kompilasi program alloc-first-fit.c melalui terminal dengan menuliskan perintah

gcc alloc-worst-fit.c -o alloc-worst-fit

```
4          14          1          15          userlinux@Linux:~$ nano
userlinux@Linux:~$ gcc alloc-worst-fit.c -o alloc-worst-fit
userlinux@Linux:~$
```

1 Nama : Dani Adrian

2 NIM : 225150201111009

- c. Jalankan program dengan menuliskan perintah **./alloc-worst-fit.exe**

```
userlinux@Linux:~$ ./alloc-worst-fit
```

Teknik Alokasi Memori - Worst Fit

Masukkan jumlah blok memori:

Open

*nama prak

1 Nama : Dani Adrian

2 NIM : 225150201111009



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

- d. Masukkan variable berikut: Jumlah proses 5, jumlah proses 4. Ukuran blok secara berurutan 15, 20, 5, 7, 4. Ukuran proses secara berurutan 2, 5, 6, 14.

```
userlinux@Linux:~$ ./alloc-worst-fit
Teknik Alokasi Memori - Worst Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14
```

- e. Capture outputnya dan simpan sebagai laporan.

```
userlinux@Linux:~$ ./alloc-worst-fit
Teknik Alokasi Memori - Worst Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14

No. Proses:    Ukuran Proses : Alokasi Blok:    Ukuran Blok:    Fragment
1              2              2              20              18
2              5              1              15              10
3              6              4              7               1
4             14              0              0
0userlinux@Linux:~$
```

- f. Amati output hasil percobaannya untuk referensi pada pembahasan.

```
No. Proses:    Ukuran Proses : Alokasi Blok:    Ukuran Blok:    Fragment
1              2              2              20              18
2              5              1              15              10
3              6              4              7               1
4             14              0              0
0userlinux@Linux:~$
```



7.5 Pembahasan

1. Berdasarkan hasil pengamatan Anda pada output setiap berkas first-fit, best-fit, dan worst-fit, maka :
 - a. Bagaimana hasil alokasi blok memori untuk setiap variable yang telah dimasukkan?

	First-fit	Best-fit	Worst-fit
15	P1: 2	P4: 14	P2: 5
20	P2: 5		P1: 2
5		P2: 5	
7	P3: 6	P3: 6	P3: 6
4		P1: 2	
Waiting (0)	P4: 14		P4: 14

a. First-fit

```
userlinux@Linux:~$ ./alloc-first-fit
Teknik Alokasi Memori - First Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14

No. Proses:   Ukuran Proses :  Alokasi Blok:   Ukuran Blok:   Fragment
1             2               1           15           13
2             5               2           20           15
3             6               4           7            1
4            14              0           0            0
userlinux@Linux:~$
```

Algoritma ini mencari lubang blok memori dari urutan pertama yang bisa diisi dengan ukuran proses yang akan dieksekusi, dengan masukan sesuai soal, algoritma first-fit menampilkan keluaran:

- Proses 1 dengan ukuran proses = 2 dialokasikan pada blok yang berukuran 15 karena lubang pertama pada blok memori berukuran 15 dan proses 1 yang berukuran 2 (kurang dari 15) cukup untuk mengisi blok memori pertama yang berukuran 15.
- Proses 2 dengan ukuran proses = 5 dialokasikan pada blok yang berukuran 20 karena lubang kedua pada blok memori berukuran 20 dan proses 2 yang berukuran 5 (kurang dari 20) cukup untuk mengisi blok memori kedua yang berukuran 20.
- Proses 3 dengan ukuran proses = 6 dialokasikan pada blok yang berukuran 7 karena lubang ketiga pada blok memori yang berukuran 5 tidak cukup menampung proses 3 yang berukuran 6



(lebih besar dari 5) sehingga beralih ke lubang berikutnya yaitu lubang keempat yang berukuran 7 dan proses 3 yang berukuran 6 cukup untuk mengisi blok memori keempat yang berukuran 7 (6 kurang dari 7).

- Proses 4 dengan ukuran proses = 14 tidak teralokasikan pada blok memori yang tersedia karena pada lubang kelima berukuran 4, proses 4 yang berukuran 14 terlalu besar untuk mengisi lubang blok memori kelima yang berukuran 4 sehingga proses 4 ditempatkan di waiting atau 0 (tidak teralokasikan).

b. Best-fit

```
userlinux@Linux:~$ ./alloc-best-fit

Teknik Alokasi Memori - Best Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14

No. Proses:   Ukuran Proses : Alokasi Blok:   Ukuran Blok:   Fragment
1             2                5              4              2
2             5                3              5              0
3             6                4              7              1
4            14                1             15              1
```

Algoritma ini mencari lubang blok memori yang terkecil namun cukup untuk bisa diisi dengan ukuran proses yang akan dieksekusi, dengan masukan sesuai soal, algoritma best-fit menampilkan output:

- Proses 1 dengan ukuran proses = 2 dialokasikan pada blok yang berukuran 4 karena lubang kelima pada blok memori berukuran 4 dan proses 1 yang berukuran 2 (kurang dari 4) cukup untuk mengisi blok memori paling kecil pertama yaitu blok memori kelima yang berukuran 4.
- Proses 2 dengan ukuran proses = 5 dialokasikan pada blok yang berukuran 5 karena lubang ketiga pada blok memori berukuran 5 dan proses 2 yang berukuran 5 cukup untuk mengisi blok memori paling kecil kedua yaitu blok memori ketiga yang berukuran 5.
- Proses 3 dengan ukuran proses = 6 dialokasikan pada blok yang berukuran 7 karena lubang keempat pada blok memori berukuran



7 dan proses 3 yang berukuran 6 (kurang dari 7) cukup untuk mengisi blok memori paling kecil ketiga yaitu blok memori keempat yang berukuran 7

- Proses 4 dengan ukuran proses – 14 dialokasikan pada blok memori yang berukuran 15 karena lubang pertama pada blok memori berukuran 15 dan proses 4 yang berukuran 14 (kurang dari 15) cukup untuk mengisi blok memori paling kecil keempat yaitu blok memori pertama yang berukuran 15.

c. Worst-fit

```
userlinux@Linux:~$ ./alloc-worst-fit

Teknik Alokasi Memori - Worst Fit
Masukkan jumlah blok memori:5
Masukkan jumlah proses:4

Masukkan ukuran blok memori:-
Blok 1:15
Blok 2:20
Blok 3:5
Blok 4:7
Blok 5:4
Masukkan ukuran proses :-
Proses 1:2
Proses 2:5
Proses 3:6
Proses 4:14
```

No.	Proses:	Ukuran Proses	: Alokasi Blok:	Ukuran Blok:	Fragment
1	2	2	20	18	
2	5	1	15	10	
3	6	4	7	1	
4	14	0	0	0	

```
userlinux@Linux:~$
```

Algoritma ini mencari lubang blok memori yang paling besar untuk bisa diisi dengan ukuran proses yang akan dieksekusi, dengan masukan sesuai soal, algoritma worst-fit menampilkan output:

- Proses 1 dengan ukuran proses = 2 dialokasikan pada blok yang berukuran 20 karena lubang kedua pada blok memori berukuran 20 dan proses 1 yang berukuran 2 (kurang dari 20) cukup untuk mengisi blok memori paling besar pertama yaitu blok memori kedua yang berukuran 20.
- Proses 2 dengan ukuran proses = 5 dialokasikan pada blok yang berukuran 15 karena lubang pertama pada blok memori berukuran 15 dan proses 2 yang berukuran 5 (kurang dari 15) cukup untuk mengisi blok memori paling besar kedua yaitu blok memori kedua yang berukuran 15.
- Proses 3 dengan ukuran proses = 6 dialokasikan pada blok yang berukuran 7 karena lubang keempat pada blok memori berukuran 7 dan proses 3 yang berukuran 6 (kurang dari 7) cukup untuk



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

mengisi blok memori paling besar ketiga yaitu blok memori keempat yang berukuran 7.

- Proses 4 dengan ukuran proses = 14 tidak teralokasikan pada blok memori yang tersedia karena pada lubang ketiga berukuran 5 dan lubang kelima berukuran 4, proses 4 yang berukuran 14 terlalu besar untuk mengisi kedua lubang blok memori tersebut yang berukuran 5 dan 4 sehingga proses 4 ditempatkan di waiting atau 0 (tidak teralokasikan).

- b. Apakah ada proses yang tidak teralokasikan pada blok memori tertentu? Jika ada, berikan penjelasan!

Ada, yaitu proses 4 pada algoritma first-fit dan worst-fit karena pada algoritma first-fit, ukuran proses 4 terlalu besar untuk mengisi blok memori terakhir yang hanya berukuran 4, kemudian pada algoritma worst-fit karena ukuran proses 4 juga terlalu besar untuk mengisi sisa blok memori yang tersedia antara lain blok memori ketiga yang berukuran 5 dan blok memori terakhir yang berukuran 4.

2. Jelaskan bagian kode program yang membuat teknik alokasi memori ini berbeda! Ulas untuk setiap teknik alokasi memori!

- a. First-fit



```
// mengambil setiap proses satu persatu
for (i = 1; i <= jmlProses; i++)
{
    for (j = 1; j <= jmlBlok; j++)
    {
        // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
        if (alokasiBlok[j] != 1)
        {
            temp = ukuranBlok[j] - ukuranProses[i];
            if (temp >= 0)
            {
                flags[i] = j;
                break;
            }
            else
            {
                temp = 0;
            }
        }
    }

    // alokasi first-fit berdasarkan informasi alokasi blok memori
    frag[i] = temp;
    alokasiBlok[flags[i]] = 1;
}
```

Mulai dari baris ke-28, mengambil setiap proses satu per satu menggunakan pengulangan for, kemudian melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses yang hasilnya diwakilkan menggunakan variabel temp. Pada algoritma ini, selama ukuran dari blok memori lebih besar dari sama dengan ukuran proses maka proses tersebut akan dialokasikan pada blok memori tersebut, kemudian lanjut mengalokasikan proses berikutnya dengan cara yang sama seperti sebelumnya hingga seluruh lubang pada blok memori telah ditelusuri, bila masih ada proses tersisa yang belum teralokasikan maka sesuai dengan keluaran, proses tersebut dialokasikan pada blok memori ke-0.

b. Best-fit



```
for (i = 1; i <= jmlProses; i++)
{
    for (j = 1; j <= jmlBlok; j++)
    {
        // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
        if (alokasiBlok[j] != 1)
        {
            temp = ukuranBlok[j] - ukuranProses[i];
            if (temp >= 0)
            {
                if (lowest > temp)
                {
                    flags[i] = j;
                    lowest = temp;
                }
            }
        }
    }

    // alokasi best-fit berdasarkan nilai fragment terkecil
    frag[i] = lowest;
    alokasiBlok[flags[i]] = 1;
    lowest = 10000;
}
```

Mulai dari baris ke-28, mengambil setiap proses satu per satu menggunakan pengulangan for, kemudian melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses yang hasilnya diwakilkan menggunakan variabel temp. Pada algoritma ini, program memeriksa keseluruhan blok memori dan mencari ukuran blok memori yang terkecil, mulai dari proses pertama bila dapat dialokasikan di blok memori terkecil yang masih kosong maka proses tersebut akan dialokasikan pada blok memori tersebut. Begitu selanjutnya ke proses-proses berikutnya hingga seluruh lubang pada blok memori telah ditelusuri, bila masih ada proses tersisa yang belum teralokasikan maka sesuai dengan keluaran, proses tersebut dialokasikan pada blok memori ke-0. Namun pada kasus kali ini karena jumlah blok memori lebih banyak dibanding jumlah proses, memungkinkan seluruh proses teralokasikan.

c. Worst-fit



```
for (i = 1; i <= jmlProses; i++)
{
    for (j = 1; j <= jmlBlok; j++)
    {
        // melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses
        if (alokasiBlok[j] != 1)
        {
            temp = ukuranBlok[j] - ukuranProses[i];
            if (temp >= 0)
            {
                if (highest < temp)
                {
                    flags[i] = j;
                    highest = temp;
                }
            }
        }
    }

    // alokasi worst-fit berdasarkan nilai fragment terbesar
    frag[i] = highest;
    alokasiBlok[flags[i]] = 1;
    highest = 0;
}
```

Mulai dari baris ke-28, mengambil setiap proses satu per satu menggunakan pengulangan for, kemudian melakukan proses alokasi dengan mengurangi ukuran blok dengan ukuran proses yang hasilnya diwakilkan menggunakan variabel temp. Pada algoritma ini, program memeriksa keseluruhan blok memori dan mencari ukuran blok memori yang terbesar, mulai dari proses pertama bila dapat dialokasikan di blok memori terbesar yang masih kosong maka proses tersebut akan dialokasikan pada blok memori tersebut. Begitu selanjutnya ke proses-proses berikutnya hingga seluruh lubang pada blok memori telah ditelusuri, bila masih ada proses tersisa yang belum teralokasikan maka sesuai dengan keluaran, proses tersebut dialokasikan pada blok memori ke-0.

3. Jelaskan fragment yang dimaksud pada tampilan output program tersebut!
Apa pengaruhnya terhadap blok memori?

Bila diperhatikan, fragment pada tampilan output merupakan hasil dari pengurangan antara ukuran blok memori yang menjadi alokasi suatu proses dengan ukuran proses tersebut. Ini menandakan bahwa fragment adalah sebuah fenomena di ruang penyimpanan yang digunakan secara tidak efisien, mengurangi kapasitas penyimpanan.

Fragment pada:

- a. First-fit



- Proses 1
 - Ukuran proses = 2
 - Ukuran blok = 15
 - Fragment = $15 - 2 = 13$ (tidak terpakai)
- Proses 2
 - Ukuran proses = 5
 - Ukuran blok = 20
 - Fragment = $20 - 5 = 15$ (tidak terpakai)
- Proses 3
 - Ukuran proses = 6
 - Ukuran blok = 7
 - Fragment = $7 - 6 = 1$ (tidak terpakai)
- Proses 4
 - Ukuran proses = 14
 - Ukuran blok pada output 0 karena tidak teralokasikan, namun berdasarkan program proses ini melalui blok memori berukuran 4
 - Maka fragment = $4 - 14 = -10$

b. Best-fit

- Proses 1
 - Ukuran proses = 2
 - Ukuran blok = 4
 - Fragment = $4 - 2 = 2$ (tidak terpakai)
- Proses 2
 - Ukuran proses = 5
 - Ukuran blok = 5
 - Fragment = $5 - 5 = 0$
- Proses 3
 - Ukuran proses = 6
 - Ukuran blok = 7
 - Fragment = $7 - 6 = 1$ (tidak terpakai)
- Proses 4
 - Ukuran proses = 14
 - Ukuran blok = 15
 - Fragment = $15 - 14 = 1$ (tidak terpakai)



c. Worst-fit

- Proses 1
 - Ukuran proses = 2
 - Ukuran blok = 20
 - Fragment = $20 - 2 = 18$ (tidak terpakai)
- Proses 2
 - Ukuran proses = 5
 - Ukuran blok = 15
 - Fragment = $15 - 5 = 10$ (tidak terpakai)
- Proses 3
 - Ukuran proses = 6
 - Ukuran blok = 7
 - Fragment = $7 - 6 = 1$ (tidak terpakai)
- Proses 4
 - Ukuran proses = 14
 - Ukuran blok = 0 karena tidak teralokasikan
 - Fragment = 0 karena bila dilihat ke dalam program, frags tertinggi adalah 0 dan karena proses 4 tidak teralokasikan maka nilai yang keluar adalah 0.

Kesimpulan

- Blok memori terdiri dari sekumpulan lubang dengan berbagai ukuran yang tersebar di seluruh memori.
- Alokasi memori adalah fenomena ketika sebuah proses tiba dan membutuhkan memori, sistem mencari lubang yang cukup ukurannya untuk mengalokasikan proses tersebut.
- Terdapat tiga teknik alokasi memori, antara lain: first-fit, best-fit, dan worst fit.
- First-fit. Alokasikan lubang pertama yang cukup besar. Pencarian dapat dimulai baik di awal set lubang atau di lokasi di mana pencarian pertama sebelumnya berakhir. Kita bisa berhenti mencari begitu kita menemukan lubang bebas yang cukup besar.



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

- Best-fit. Alokasikan lubang terkecil yang cukup besar. Kita harus mencari seluruh daftar lubang yang tersedia, kecuali daftar lubang itu diurutkan berdasarkan ukuran. Teknik ini menghasilkan lubang sisa terkecil.
- Worst-fit. Alokasikan lubang terbesar. Sekali lagi, kita harus mencari seluruh daftar lubang, kecuali jika diurutkan berdasarkan ukuran. Teknik ini menghasilkan lubang sisa terbesar, yang mungkin lebih berguna daripada lubang sisa yang lebih kecil dari pendekatan best-fit.
- Mudahnya, first-fit mengalokasikan proses ke dalam blok memori mulai dari lubang pertama dan selama ukuran proses lebih kecil sama dengan dari blok memori yang sedang diakses maka proses akan dialokasikan pada blok memori tersebut, bila tidak akan berlanjut mencari ke blok memori selanjutnya, dan bila sudah di akhir blok memori proses masih belum dialokasikan maka proses tersebut dianggap tidak teralokasikan,
- best-fit mengalokasikan mulai dari mencari blok memori yang terkecil hingga blok memori yang terbesar, dan
- worst-fit mengalokasikan mulai dari mencari blok memori yang terbesar hingga blok memori yang terkecil.