Ivan Andrianto

13513039

7.3.

a. Need matrix

A B C D

P0 0 0 0 0

P1 0 7 5 0

P2 1 0 0 2

P3 0 0 2 0

P4 0 6 4 2

b. Ya. Terdapat beberapa sequence (urutan) proses yang memenuhi. Contoh safe sequence: < P0, P2, P1, P3, P4>

c. Karena available nya (1,5,2,0), alokasi dapat dilakukan. Kemudian available nya menjadi (1,1,0,0). Alloc P1 menjadi (1,4,2,0) dan Need P1 menjadi (0,3,3,0). Contoh safe sequence: <P0, P2, P3, P1, P4>

7.4. Kedua algoritma bisa digunakan untuk mencegah deadlock. Tetapi, Circlular-Wait lebih efisien karena memungkinkan beberapa proses untuk menggunakan resource selama tidak terjadi deadlock. Sedangkan, Containment hanya memungkinkan setiap resource digunakan untuk satu proses pada suatu waktu. Tetapi, Containment memiliki kelebihan yaitu lebih mudah diterapkan daripada Circular-Wait sehingga untuk proses yang tidak dapat diperkirakan berapa banyak resource yang akan digunakan, akan sulit menerapkan Circular-Wait.

7.5. Pada safety algorithm, terdapat vector Work dan Finish dengan panjang masing-masing m dan n.

(i) Mula-mula dinisialisasi dengan Work=Available dan Finsih[i]=false untuk i=0….n-1.

(ii) Kemudian cari Finish[i] yang bernilai false (kompleksitas n) dan Needi <= Work (kompleksitas m).

(iii) Jika terdapat nilai i yang memenuhi, Work = Work + Allocation, ubah Finish[i] menjadi true, dan ulangi langkah (paling banyak terdapat n kasus seperti ini) (kompleksitas n)

(iv) Jika finish[i]== true untuk setiap nilai i, berarti sistem berada pada safe state.

Dengan demikian, dapat disimpulkan bahwa kompleksitasnya O(m x n2)

7.6. a. Alasan untuk menggunakan deadlock avoidance adalah bisa memastikan deadlock tidak mungkin terjadi. Meskipun turnaround time meningkat, semua job dapat berjalan terus menerus.

b. Alasan untuk tidak menggunakan deadlock adalah waktu turnaround yang lebih rendah, frekuensi terjadinya deadlock yang rendah, dan cost dari terjadinya deadlock yang rendah.