Analisis Deadlock dan Algoritma Bankir

1. Analisis Cara Kerja Deadlock dengan Algoritma Bankir
Deadlock terjadi ketika sekumpulan proses saling menunggu sumber daya yang tidak pernah dilepaskan. Algoritma Bankir digunakan untuk mencegah deadlock dengan memastikan sistem selalu dalam keadaan aman (safe state) sebelum mengalokasikan sumber daya baru. Langkah kerja algoritma Bankir: 1. Setiap proses harus menyatakan jumlah maksimum sumber daya yang dibutuhkan. 2. Sistem memeriksa apakah permintaan sumber daya dari suatu proses masih memungkinkan sistem berada pada safe state. 3. Jika aman → sumber daya diberikan. Jika tidak aman → proses harus menunggu. 4. Setelah proses selesai, sumber daya dikembalikan ke sistem. Dengan cara ini, Deadlock tidak akan terjadi, karena sistem hanya mengizinkan alokasi ketika keadaan aman.

2. Perbedaan Teknik Grafik Alokasi Sumber Daya dan Algoritma Bankir
AspekTeknik Grafik Alokasi Sumber DayaAlgoritma Bankir PendekatanMenggunakan graf untuk memvisualisasikan hubungan antara proses dan sumber daya Menggunakan perhitungan numerik (matriks) untuk menentukan keadaan aman Deteksi vs PencegahanDigunakan untuk mendeteksi atau menganalisis deadlock Digunakan untuk mencegah deadlock KelebihanMudah dipahami secara visualLebih efisien untuk sistem besar KekuranganSulit diterapkan pada sistem dengan banyak proses/sumber daya Membutuhkan banyak perhitungan dan data (max, need, allocation, available)

3. Konsep yang Paling Efektif dan Alasannya
Algoritma Bankir lebih efektif digunakan dalam sistem komputer modern, karena: Dapat mencegah deadlock sebelum terjadi. Cocok untuk sistem multiprogramming yang kompleks. Dapat dijalankan secara otomatis oleh sistem operasi tanpa analisis manual seperti grafik.

4. Contoh Program C: Algoritma Bankir

```
#include <stdio.h>

int main() {
    int n, m;
    int alloc[10][10], max[10][10], avail[10];
    int need[10][10], finish[10], safeSeq[10];
    int i, j, k, count = 0;

    printf("Masukkan jumlah proses: ");
    scanf("%d", &n);
    printf("Masukkan jumlah jenis sumber daya: ");
    scanf("%d", &m);

    printf("Masukkan matriks ALLOCATION:\n");
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            scanf("%d", &alloc[i][j]);

    printf("Masukkan matriks MAX:\n");
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            scanf("%d", &max[i][j]);

    printf("Masukkan vektor AVAILABLE:\n");
    for(i=0;i<m;i++)
        scanf("%d", &avail[i]);

    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            need[i][j] = max[i][j] - alloc[i][j];

    for(i=0;i<n;i++)
        finish[i] = 0;

    while(count < n) {
        int found = 0;
        for(i=0;i<n;i++) {
```

```
            if(finish[i] == 0) {
                int flag = 0;
                for(j=0;j<m;j++) {
                    if(need[i][j] > avail[j]) {
                        flag = 1;
                        break;
                    }
                }
                if(flag == 0) {
                    for(k=0;k<m;k++)
                        avail[k] += alloc[i][k];
                    safeSeq[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }
        if(found == 0) {
            printf("\nSistem tidak berada dalam keadaan aman (Deadlock mungkin terjadi)\n");
            return 0;
        }
    }

    printf("\nSistem berada dalam keadaan AMAN.\nUrutan proses yang aman: ");
    for(i=0;i<n;i++)
        printf("P%d ", safeSeq[i]);
    printf("\n");
    return 0;
}
```

Analisis Program:
Program di atas menggunakan Algoritma Bankir untuk memastikan sistem berada pada keadaan aman. Langkah-langkah yang dilakukan: Input data jumlah proses, sumber daya, matriks **Allocation**, **Max**, dan **Available**. Menghitung matriks **Need** = Max - Allocation. Melakukan pengecekan setiap proses apakah kebutuhan (Need) masih bisa dipenuhi dengan sumber daya yang tersedia. Jika bisa, proses dijalankan dan sumber daya dikembalikan ke sistem. Jika semua proses selesai, sistem berada dalam **safe state**. Jika ada proses yang tidak dapat dijalankan, sistem **tidak aman (unsafe)** dan kemungkinan terjadi deadlock.