1. **Jelaskan tahapan dalam algoritma binary search!, serta implementasi binary search berikut ini A = {3, 7, 9, 13, 20, 27, 37, 57} untuk mencari nilai 20!.**

   Algoritma Binary Search bekerja dengan menghapus setengah dari data setiap kali diperiksa. Data harus diurutkan sebelum pencarian biner dapat dilakukan. Dengan konsep seperti itu, ruang lingkup pengambilan data menjadi lebih kecil setiap kali proses iteratif/perulangan dilakukan.

```
A = {3, 7, 9, 13, 20, 27, 37, 57};
// Data terendah adalah L = 0
// Data tengah adalah M = 3
// Data tertinggi adalah H = 7

A = {13, 20, 27, 37, 57};
// Setengah data pertama dihapus
// Data terendah adalah L = 3
// Data tengah adalah M = 5
// Data tertinggi adalah H = 7

A = {13, 20, 27};
// Setengah data kedua dihapus
// Data terendah adalah L = 3
// Data tengah adalah M = 4
// Data tertinggi adalah H = 5

A = {20, 27};
// Setengah data pertama dihapus
// Data terendah adalah L = 3
// Data tengah adalah M = 3
// Data tertinggi adalah H = 4
// Data sudah ditemukan
```

2. **Bagaimana membedakan algoritma sorting jenis Comparison Sort dan Non-Comparison Sort? Selesaikan kasus berikut ini dengan menggunakan radix sort!**

1	2	3	4	5	6	7	8	9	10	11	12	13	14
27	177	48	168	32	626	214	58	79	10	47	335	48	13

   Semua masalah pengurutan yang dapat diurutkan dengan algoritma *non-comparison sort* dapat diurutkan dengan algoritma *comparison sort*, tetapi tidak sebaliknya.

   Algoritma Comparison Sort mengurutkan item dengan membandingkan nilainya satu sama lain. Ini dapat diterapkan pada situasi penyortiran apa pun.

   Algoritma Non-Comparison Sort menggunakan karakter internal dari nilai yang akan diurutkan. Itu hanya dapat digunakan dalam situasi tertentu dan membutuhkan nilai-nilai tertentu.

```
Array[] = {27, 177, 48, 168, 31, 626, 214, 58, 79, 10, 47, 335, 48,
13};
// Urutkan angka paling belakang terlebih dahulu
```

```
Array[] = {10, 31, 13, 214, 335, 626, 27, 177, 47, 48, 168, 58, 48,
79};
// Selanjutnya urutkan angka kedua dari belakang

Array[] = {10, 13, 214, 626, 27, 31, 335, 47, 48, 48, 58, 168, 177,
79};
// Selanjutnya urutkan angka ketiga dari belakang

Array[] = {10, 13, 27, 31, 47, 48, 48, 58, 79, 168, 177, 214, 335,
626};
// Array sudah terurut
```

3. **Bagaimana membedakan algoritma Binary Tree dan Skewed Tree? serta Inorder Traversal, Preorder Traversal dan Postorder Traversal dalam Binary Tree Traversal!.**
   - ***Binary Tree*** adalah pohon biner dengan syarat setiap simpul hanya boleh memiliki paling banyak dua *child*, dan kedua *child* tersebut harus terpisah.
   - ***Skewed Tree*** adalah pohon biner di mana semua simpul hanya memiliki satu *child* atau tidak ada *child*. Ada 2 jenis *skewed tree*, yaitu *Left Skewed* (condong ke kiri) dan *Right Skewed* (condong ke kanan).
   - ***Inorder Traversal*** melintasi *subtree* kiri, kemudian mengunjungi akar (*root*), dan kemudian melintasi *subtree* kanan.
   - ***Preorder Traversal*** pertama-tama mengunjungi akar (*root*), lalu melintasi *subtree* kiri, lalu *subtree* kanan.
   - ***Postorder Traversal*** pertama melintasi *subtree* kiri, kemudian *subtree* kanan, dan kemudian mengunjungi akar (*root*).

4. **Paparkan dengan jelas dan singkat tentang graph dan bagaimanakah perbedaan antara undi-graph dan di-graph serta gambarkan penerapan graph dalam kasus kehidupan sehari-hari yang Anda dapat lihat dan gunakan!**

   Graf adalah himpunan verteks atau node yang dihubungkan oleh sisi (atau edge atau arc). Biasanya, graf digambarkan sebagai kumpulan titik (mewakili verteks) yang dihubungkan oleh garis (mewakili sisi).

   Di-Graph adalah himpunan verteks (node) yang dihubungkan oleh sisi, setiap simpul memiliki arah yang terkait dengannya. Sisi biasanya diwakili oleh panah yang menunjuk ke arah grafik yang dapat dilalui.

   Undi-Graph adalah himpunan verteks (node) yang sisi-sisinya terkait tanpa arah. Oleh karena itu, grafik ini dapat dilalui di kedua arah. Tidak adanya panah menunjukkan bahwa grafik tidak berarah.

5. **Dept. IT mempunyai 6 kelompok kerja yang setiap bulannya masing-masing selalu mengadakan rapat satu kali. Keenam kelompok kerja dengan masing-masing anggotanya adalah: K1 = {Amir, Budi, Yanti}, K2 = {Budi, Hasan, Tommy}, K3 = {Amir, Tommy, Yanti}, K4 = {Hasan, Tommy, Yanti}, K5 = {Amir, Budi}, K6 = {Budi, Tommy, Yanti}. Berapa banyak waktu rapat berbeda yang harus direncanakan sehingga tidak ada anggota kelompok kerja yang dijadwalkan rapat pada waktu yang sama. Gambarkan graph yang merepresentasikan persoalan ini lalu (edge/sisi menyatakan apa, vertex/simpul menyatakan apa) tentukan jumlah waktu rapat ini.**

   K1 = {Amir, Budi, Yanti}

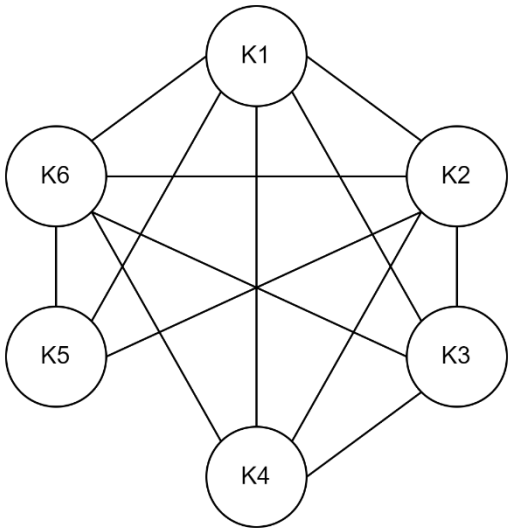
   K2 = {Budi, Hasan, Tommy}

   K3 = {Amir, Tommy, Yanti}

   K4 = {Hasan, Tommy, Yanti}

   K5 = {Amir, Budi}

   K6 = {Budi, Tommy, Yanti}



   1 waktu untuk K1

   1 waktu untuk K2

   1 waktu untuk K3

   1 waktu untuk K4 dan K5

   1 waktu untuk K6

   Jumlah Waktu untuk Rapat = **5**.