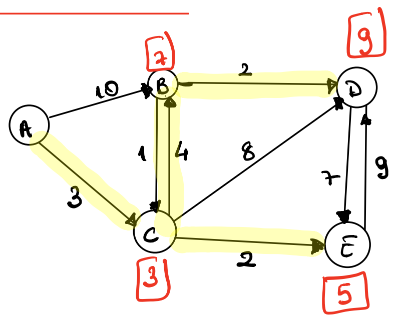
Dijkstra Algoritması;



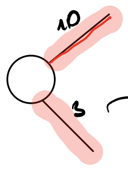
Q	A	B	C	D	E
Extract min	0	∞	∞	∞	∞
	–	10	decrease $\boxed{3}$ key	∞	∞
	–	4+3 7	–	8+3 11	3+2 $\boxed{5}$
	–	$\boxed{7}$	–	11	–
	–	–	–	$\boxed{9}$	–

⇒ ilk adım başlangıç 0 diğerleri ∞

→ decrease key

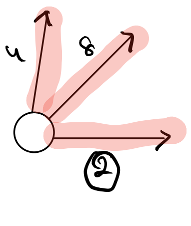
$S = \{ A\ C\ E\ B\ D \}$

1. adım: A dan ayrılan tüm kenarları gez et



→ en azı çıkar (Extract min)

2. adım. C den ayrılan tüm kenarları gezet



En azı çıkar (2)

3. adım. E ...

Algoritma:

```
void Dijkstra (int weight [] [WSIZE], int baslangic)
    {   int yol [WSIZE];
        int perm [WSIZE];
        // ilk aşamada ziyaret edilmemiş bütün düğümlere sonsuz atanır
        // Ayrıca ziyaret edilmediklerini gösterir NONMEMBER değeri at.
        for ( int i=0; i<WSIZE; i++)
        { yol[i]= INT_MAX;
          perm[i] = NOMEMBER; }
      // Başlangıç düğümünün uzaklığı
             yol[baslangic] = 0;
   for ( int dolas = 0;   dolas < wSIZE-1; dolas++)
        {     int min = INT_MAX
              int min_index;
      // Hiç uğramamış düğümlere değer ata
      // İçlerinden en küçüğü seç
        for  (int j=0 ;  j<wSIZE;  j++)
              if (perm[j] == NOMEMBER && yol[j] <= min)
                     min = yol[j];  min_index = j;
      // Yukarıdaki adım sonrası en küçük düğüm
                  int u = min_index;
      perm[u]= MEMBER;
              // Relax işlemi
        for (int v=0; v<wSIZE; v++)
                    if ( ! perm[v] && weight[u][v] && yol[u]!=INT_MAX
                         && yol[u]+weight[u][v] < yol[v]) // Gezilmiş Düğümler
                          yol[v] = yol[u]+ weight[u][v] // daha kısa yol buldaysan
                                                           min yolu update et.
}
         for (int i=0; i <wSIZE ; i++)
              printf ("başlangıç düğümü %d bitiş %d en kısa yol %d\n",
           başlangıç, i, yol[i]);  //Düğümler ve uzaklıklar yazdır
```