PARALEL

BİLGİSAYARLAR

PROJE RAPORU

Amaç

Needleman - Wunsch Algoritması kullanılarak Global Hizalama yapan seri ve paralel iki yazılımın hazırlanması.

Needleman - Wunsch Algoritması

Algoritma, Protein ve DNA sekanslarının hizalanması için bulundu. 1970 yılında büyük dizilerin hizalanması için ilk defa kullanılan ilk Dinamik Programdır. İki dizi içindeki benzerlikler ve benzer olmayan durumlar puanlandırılır ve bir tabloya yerleştirilir. Puanlama işlemi bittikten sonra sağ alt köşeden başlayarak en az penaltı puanına sahip olan yol optimum hizalamayı verir.

Elimizde

GCATGCU

GATTACA

şeklinde iki adet nükleotid dizisi olsun. Bu iki dizinin onlarca hizalanma ihtimali mevcut fakat bize en optimum olanı lazım. Bunun için dizi elemanlarını yatay ve dikey olarak tabloya yerleştiriyoruz. Tablonun üzt kısmında bir satır boş bırakılıyor.

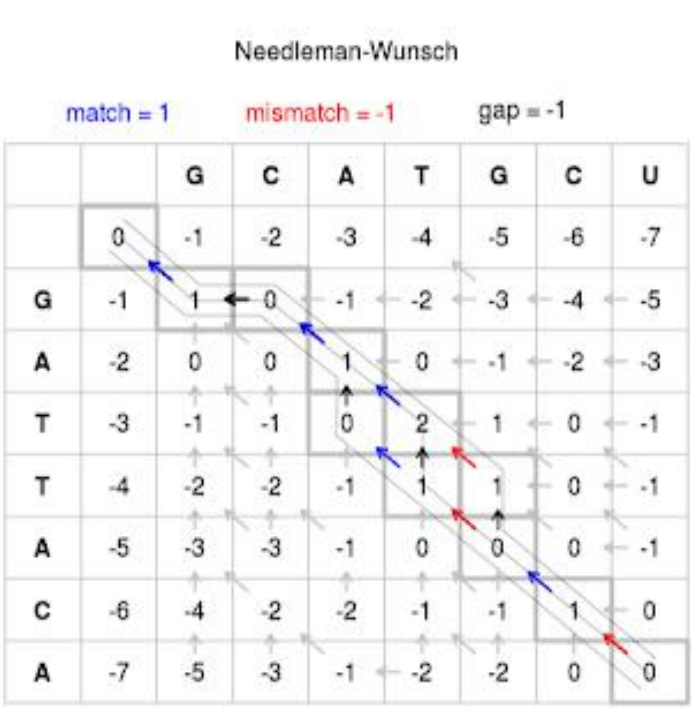
		G	C	A	T	G	C	U
G								
A								
T								
T								
A								
C								
A								

Daha sonra bu tabloyu en üst satırdan başlayarak dolduruyoruz. Birbirinin aynı olan harfler (G/G gibi) +1 ile uyumsuzlar (G/A gibi) -1 ile boşluk harf kombinasyonları (G/"-" gibi) -1 ile puanlandırılsın.

Bu algoritmada komşu olan karelerden en yüsek değerli olanı alınır ve puanlama onun üzerine eklenerek yapılır. İlk satırı doldurduğu- muzda aşağıdaki gibi olacaktır;

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1							
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

Görüldüğü gibi en üstteki sıra 0 dan başlayarak ceza puanı olan -1 eklenerek devam etti. Yatay ve dikey sıralartamandık tan sonra G ile başlayan ilk yatay sıraya geliyor. Burada G-G uyumu +1 ile puanlandıracak, komşu karelerde en büyük değer 0 olduğu için + eklenecek ve o kare +1 olacak;
Şimdi ikinci satırdaki G-C için -1 değeri vermeliyiz. Komşu karelerdeki en büyük değer +1 dolayısı ile -1 ile toplandığında bu karedeki değer 0 olacaktır;
Bu şekilde tüm tabloyu dolduruyoruz.



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Bu tabloda çeşitli optimum hizalamalar mevcut, mavi ve kırmızı oklarla köşegen geçişler harf karşılıklarını belirtirken yatay ve dikey geçişler (siyah okla belirtilenler) "-" indel boşluk karakterini ifade eder.
Projede skor değeri olarak köşe değeri yani [8][8] hücresi kullanılmıştır.Verilen sekanslar 200'er bazdan oluştuğundan [200][200] hücresi kullanılmıştır.
Kullanılan yazılıma hız kazandırmak amacı ile OpenMP kullanılmıştır.

OpenMP

OpenMP Solaris, IBM AIX , HP-UX, GNU/LINUX, MAC OS X ve Windows işletim sistemleri üzerinde çoğu işlemci mimarisi üzerinde Fortran, C++, C programlama dillerinde çoklu platform paylaşımlı bellek çoklu işlemeyi destekleyen bir uygulama geliştirme arayüzüdür, yani bir API'dir. OpenMP derleyici yönergelerinin kütüphane rutinlerini ve ortam değişkenlerinin çalışma zamanı davranışını etkileyen bir kümesini içerir. OpenMP çoklu iş parçacığı gerçekleştirimidir. Çoklu iş parçacığı ana iş parçacığının(sırasıyla yürütülen komutların bir dizisi) belirli bir sayıda yardımcı iş parçacıklarını durdurması ve bir görev onlar arasında paylaştırması olan paralelleştirme metodudur. İş parçacıkları birbiri

ardında paralel şekilde çalışırlar. Farklı işlemcilerin iş parçacıkları farklı çalışma zamanı ortamlarını kendilerine tahsis ederler.

Visual studio da OpenMP kullanabilmek için önce ayarlar menüsünden OpenMP desteğini aktifleştirmek gerekir.Ardından `<omp.h>` **header dosyası include edilmelidir.**

TASARIM

Öncelikle fasta dosyadan sekanslar okunmuştur. Okunan veri bir String dizisinde karşılaştırmak üzere tutulmuştur.
Needleman - Wunsch Algoritması ışığında gerekli hesaplamalar yapılıp skor değerleri bulunmuştur.
5000 sekans olduğundan 5.000 x 4.999 / 2 = 12.497.500 sunuç bulunmuştur.
12.497.500 float değer C++ da bulunan dizi[ ] veri yapısına sığmadığı gözlemlenmiş ve aynı zamanda en büyük skor değerleriyle bu skor değerlerinin üretildiği sekanslar da istendiğinden bağlı liste veri yapısı kullanılmıştır.

```
struct node {
float deger;
node* next;
int sec1;
int sec2;
};
```

Üretilen her skor büyükten küçüğe sıralı bir şekilde bağlı listeye eklenmiş ve en büyük 20 skor değeri gösterilmiştir.

Programda matris, toplama, çıkarma, atama,sıralama işlerinin yoğun olduğu yer paralelleştirilmiştir.

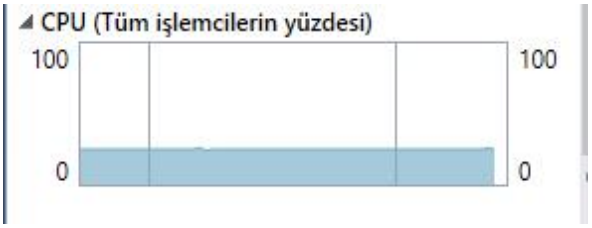
```
#pragma omp for nowait
   for (int i = 0; i < 200; i++) {
      for (int j = i + 1; j < 200; j++) {
                    .
                    .
                    .
                    .

for (int i = 1; i < 201; i++) {
      for (int j = 1; j < 201; j++) {
         dizi[i][j] = sirala(dizi[i - 1][j - 1] +
dizi[i][j], dizi[i - 1][j] + gap, dizi[i][j - 1] + gap);
```

Gibi yerlerde OpenMP nin for komutunu bölen #pragma omp for nowait kullanılmıştır. Farklı thread sayıları ile denemeler yapılmıştır.

PERFORMANS TESTLERİ

SERİ:



500 sekans : 187.552 sn.
(3,1dk)

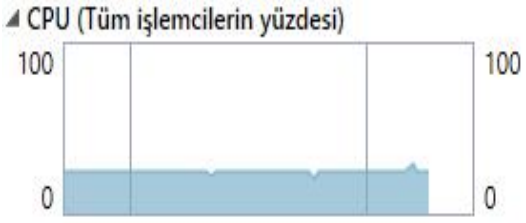
Skor	s1	s2
718.198	1	14
716.985	6	31
712.125	10	20
710.911	30	42
709.698	24	25
703.625	40	51
702.412	8	13
696.339	3	4
692.692	19	22
691.479	0	2
307.716	200	325

600 sekans : 380.247 sn. (6,3dk)

700 sekans : 827.951 sn. (13,7dk)

800 sekans : 1703.54 sn. (28,3dk)

900 sekans :3228.17 sn. (53,8dk)



1000 sekans : 4236.38 sn. (70,6 dk.)

Skor	s1	s2
718.198	1	14
716.985	6	31
712.125	10	20
710.911	30	42
709.698	24	25
703.625	40	51
702.412	8	13
696.339	3	4
692.692	19	22
691.479	0	2
319.855	652	768
312.561	258	860
310.127	573	973
307.716	200	325
307.708	634	667
307.701	490	515
306.495	543	554
306.495	437	798
306.495	567	698
306.495	301	490

Zamanın logaritmik bir şekilde arttığı gözleniştir 5k da çok yüksek rakamlara denk geldiğinden seride deneme yapılmamıştır

PARALEL: 10 Thread

500 sekans : 30.911 sn.

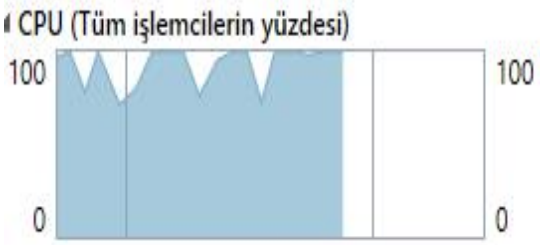
600 sekans : 47.63 sn.

700 sekans : 78.325 sn. (1,3dk)

800 sekans : 123.242 sn. (2 dk)

900 sekans : 190.818 sn. (3,1dk)

1000 sekans : 265.752 sn. (4,4dk)



Skor s1 s2
718.198 1---14
716.985 6---31
712.125 10---20
710.911 30---42
709.698 24---25
703.625 0---51
702.412 8---13
696.339 3---4
692.692 19---22
691.479 0---2

2000 sekans : 4283.2 sn. (71,3dk)

PARALEL: 20 Thread

500 sekans : 28.902 sn.

600 sekans : 36.441 sn.

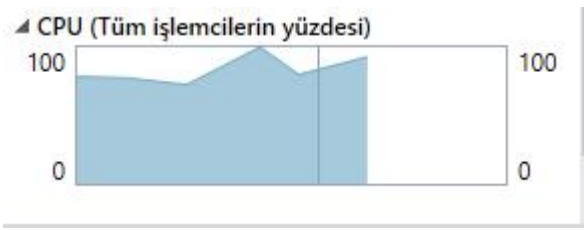
700 sekans : 59.019 sn.

800 sekans : 193.207 sn. (3,2dk)

900 sekans : 258.69 sn. (4,3dk)

1000 sekans : 172.213sn. (2,8dk)

2000 sekans : 2041.74 sn. (34dk)



PARALEL: 30 Thread

500 sekans : 24.055 sn.

600 sekans : 34.818 sn.
700 sekans : 49.894 sn.
800 sekans : 62.853 sn. (1dk)
900 sekans : 87.023 sn. (1,45dk)
1000 sekans : 117.677 sn. (1,95dk)
2000 sekans : 1261.62 sn. (21dk)
3000 sekans 6019.57sn (1000,3dk)

Skor	s1 s2
718.198	1--14
716.985	6--31
712.125	10--20
710.911	30--42
709.698	24--25
703.625	0--51
702.412	8--13
696.339	3--4
692.692	19--22
691.479	0--2
318.641	48--1332
310.135	95--1257
310.135	49--2821