Kor. J. Appl. Microbiol. Biotechnol.
Vol. 25, No. 6, 560-565 (1997)

DNA 염기 서열의 단편 조립 프로그램 개발

이병욱[1,2] · 박기정[1] · 박 완[2] · 박용하[1]*
[1]한국과학기술연구원 생명공학연구소, [2]경북대학교 미생물학과

Development of a Program for Fragment Assembly from DNA Sequence Data. Byung-Uk Lee[1,2], Kie-Jung Park[1], Wan Park[2] and Yong-Ha Park[1]*. *[1]Korea Research Institute of Bioscience and Biotechnology, KIST, Taejon 305-600, Korea, [2]Department of Microbiology, Kyungpook National University, Taegu 702-701, Korea* – DNA fragment assembly is a major concern in shot-gun DNA sequencing project. It is to reconstruct a consensus DNA sequence from a collection of random oritented fragments. We developed a computer program that is useful for DNA fragment assembly. Inputs to the program are DNA fragment sequences including IUB-IUPAC bases. The program produces the most probable reconstruction of the original DNA sequence as a text format or a PostScript format. The program consists of four phases: the first phase quickly eliminates fragment pairs that can not possibly overlap. In the second phase, the quality of overlap between each pair is calculated to a score. In the third phase, overlap pairs are sorted by their scores and consistency of the overlaps is checked. The last phase determines consensus sequences and displays them. The performance of fragment assembly program was tested on a set of DNA fragment sequences which were generated from long DNA sequences of GenBank by a fragmentation program.

DNA sequencing 프로젝트에서 매우 긴 DNA 염기 서열을 밝혀내려는 경우, 우선 그 DNA 가닥을 여러 개의 DNA 단편(fragment)들로 만든 후, 각 단편들의 염기 서열을 알아낸다. 그리고 염기 서열이 밝혀진 단편들로부터 본래의 긴 DNA 염기 서열을 재구성한다. 이 경우 발생하는 재구성 문제를 'contig 구성 문제'(1)라 하며, 궁극적으로 이 문제는 그 자체의 복잡성과 그로 인한 많은 계산량 때문에 컴퓨터의 빠른 계산 능력을 필요로 한다.

현재까지 단편 염기 서열로부터 contig를 구성하는 프로그램으로는 SEQAID(SEQencing AID)(2), CAP (Contig Assembly Program)(3) 등이 알려져 있다. 이들 대부분 프로그램의 입력 단편에 사용할 수 있는 염기는 A, C, G, T이며, 의미가 모호한 염기에 대해서는 N으로 나타낸다. 실제 sequencing 실험에서 의미가 불확실한 염기에 대해서도, purine인 경우, pyrimidine인 경우, 혹은 A를 제외한 나머지 염기 중 하나인 경우 등과 같이, 어느 정도의 정보를 가진 경우가 있다. 이들 각각의 경우를 단순히 N으로만 나타내면, 이러한 정보를 이용할 수 없으며 또한 잘못된 contig를 구성하는 원인이 되기도 한다. 이런 정보들을 입력 단편 염기로 표현하기 위해서는 IUB-IUPAC(International Union of Biochemistry-International Union of Pure and Applied Chemistry) 염기 표기법(4)을 사용하여야 하며, 또한 이들 IUB-IUPAC 염기를 처리할 수 있는 기능을 가진 프로그램의 개발도 요구된다.

대부분의 contig 구성 프로그램들은 텍스트 사용자 환경만을 지원한다. 컴퓨터 소프트웨어의 발달로 현재는 사용자들에게 그래픽 환경을 제공하는 다양한 그래픽 사용자 환경(Graphic User Interface:GUI)이 개발되어 있다. 그래픽의 사용은 컴퓨터 사용에 익숙하지 않은 사용자들에게 친숙한 환경을 제공하여, 그들이 쉽게 프로그램을 사용할 수 있게 한다. 기존 프로그램들은 결과 내용을 주로 텍스트로 출력하는데, 프로그램이 결과 내용 정보를 효과적으로 제공하기 위해서는 다양한 정보를 그래픽 형식으로 표현할 필요가 있다.

본 연구에서는 기존에 알려진 프로그램들의 단점을 보완하는 contig 구성 프로그램인 FAP(Fragment Assembly Program)를 개발하였다. 본 프로그램은, 입력으로 IUB-IUPAC 염기 DNA 단편들을 사용하고, 사용자 환경을 개선하였으며, 출력 양식으로 텍스트와 포스트스크립트(PostScript) 양식(5)을 모두 지원하도록 하였다.

FAP의 테스트 데이터를 얻기 위해서 GenBank 데이타베이스(database)에서 일정한 길이의 sequence들을 추출하여, 단편화(fragmentation) 프로그램을 사용하여, 적당한 크기 단편으로 자르고 각각의 단편에 임의의 mutation을 유발시킨 후, 2~3개의 임의의 염기를 선정하여 그 염기를 포함하는 모호한 염기로 전환된 IUB-IUPAC 염기를 사용하였다. 이들에 대해 본 프로그램을 수행시킨 결과, 모두 original sequence를 구성하였으며,

*Corresponding author
Tel. 82-42-860-4620, Fax. 82-42-860-4625
E-mail: yhpark@mail.kribb.re.kr
Key words: DNA fragment, Conting, IUB-IUPAC base, DNA assembly program

실행 시간은 단편의 수에 비례하는 것을 알 수 있었다.

재료 및 방법

개발 환경

FAP는 C 언어로 짜여진 1개의 소스 화일로 구성되었으며, UNIX 워크스테이션인 INDIGO 2(운영체계:IRIX 5.3, CPU:MIPS R4400, main memory:64 Mbytes)에서 개발하였다. 본 프로그램은 표준 C 라이브러리와 Open Software Foundation(OSF)에서 개발한 Motif(release 1.2)를 사용하여 사용자 그래픽 인터페이스를 구성하였다.

알고리즘의 개요

본 프로그램의 알고리즘은 3단계로 구성되었다. 첫 번째 단계는 모든 가능한 단편 쌍들 중에서 중첩 가능성이 있는 단편 쌍들을 선별하는 단계로서, 단편 쌍의 최소 수용 중첩 길이(minimum acceptable overlap length)를 기준으로 사용하였다. 두 번째 단계는 선별된 단편 쌍들의 중첩 정도를 측정하는 단계로서, 각 단편 쌍들의 최대치 정렬(maximum score alignment)을 사용하였다. 마지막 단계에서는, 각 단편 쌍들을 중첩 정도를 기준으로 정렬한 후, 중첩 정도가 큰 단편 쌍부터 연결하여 그 결과를 연결 목록(linked list)에 저장하여 contig를 구성하였다.

중첩 가능성 있는 단편 쌍 선별

가능한 모든 단편 쌍들에 대하여 각 단편 쌍들이 일정한 길이 만큼의 중첩을 형성할 수 있는지를 측정해서, 중첩을 형성할 수 없는 단편 쌍들을 다음 단계인 정렬(alignment) 계산 과정에서 배제시켜 전체 프로그램의 실행 시간을 감소시킬 수 있다.

이러한 기능을 수행하기 위해서 최소 수용 중첩 길이 개념을 이용하였는데, 그 길이는 단편들의 평균 길이의 0.15~0.20배가 되어야 한다(6). 본 프로그램에서는 이 방법을 구현하기 위해서 해싱(hashing)(7)을 사용하였다. 즉, 각 단편의 5′ 말단 염기에서부터 한 염기씩 증가시키면서 일정 길이의 염기열에 대한 해시 값으로 해당 단편의 해시 테이블(hash table)을 구성한다. 이 경우 만약 해당 염기가 A, C, G, T 이외의 IUB-IUPAC 염기이면, 이를 A, C, G, T의 조합으로 전환한다. 모든 단편들의 해시 테이블이 만들어지면, 각 단편 쌍에 대한 중첩을 검사한다. 한 단편의 5′ 말단 염기에서 한 염기씩 3′ 말단 방향으로 나가면서, 해시 테이블을 참조하여 두 단편이 공통 염기열을 가지는지를 검사하여 공통 염기열의 길이를 더한다(Fig. 1). 두 단편의 공통 염기열의 길이가 최소 수용 중첩 길이와 같거나 긴 경우, 그 단편 쌍은 다음 단계로 넘어가고, 만약 단편의 3′ 말단까지의 공통 염기열의 길이가 최소 수용 중첩 길이보다 짧은 경우, 두 단편 쌍은 중첩을 형성할 가능성이 없는 것으로 간주되어 다음 단계로 넘어갈 수 없다.

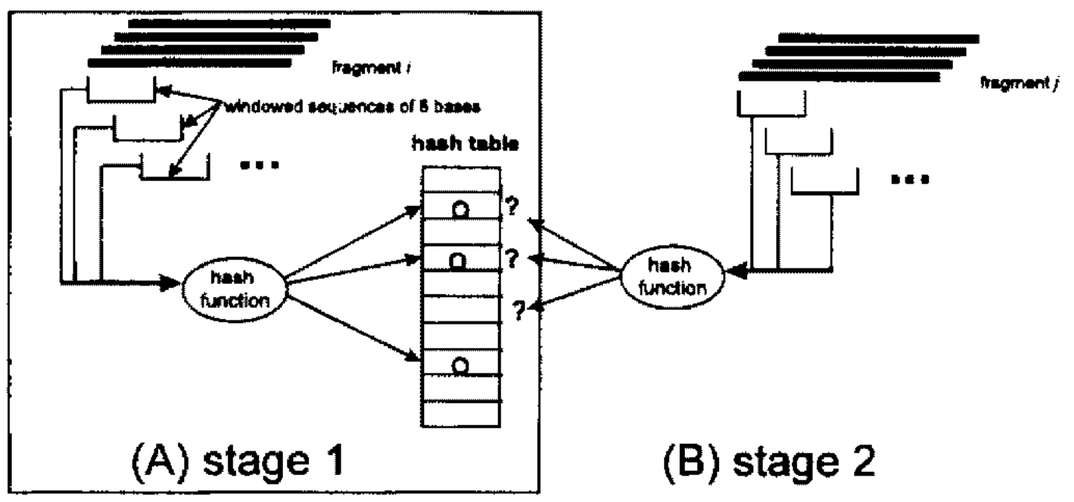

Fig. 1. Total process to select fragment pairs, which can possibly overlap.
(A) stage 1 : A hash table of fragment *i* is constructed from hash values for windowed sequences of 6 bases, which are from fragment *i*. Each hash value is produced by the hash function. The window advances one base from 5'end to 3'end of fragment *i*.
(B) Stage 2 : The windowed sequences of fragment *j* are converted to their hash values by the hash function and while the hash values of contiguous windowed sequences form fragment *j* are matched with those from fragment *i*, 6 is added to the overlap length of fragment pair (*i,j*). The hash functions in stage 1 and stage 2 are the same.

선별된 단편 쌍의 최대치 정렬

다음 단계는 최소 수용 중첩 길이 이상 중첩되는 각 단편 쌍들의 최대치 정렬을 찾는 과정으로, 본 프로그램에서는 Smith-Waterman 알고리즘(8)을 사용하였다. Contig 구성의 경우는 각 단편 쌍의 두 단편 사이의 정렬 방식으로 4가지 형태가 나올 수 있다(Fig. 2). 즉, 중첩 부분은 항상 두 단편 중 한 단편의 5′ 말단에서 시작하여 한 단편의 3′ 말단에서 끝나게 된다. 이러한 contig구성의 특수한 조건을 충족시키기 위해서 아래의 식을 사용하여 각 단편 쌍의 최대치를 구하였다. 이 식에서 N, M은 두 단편의 길이이며, H(i, j)는 한 단편의 i번째 염기와 다른 단편의 j번째 염기로 끝나는 정렬의 최대치이다.

$$\text{최대치}=\max\{H(i, N), H(M, j)\},\ 1\leq i\leq M \text{ and } 1\leq j\leq N$$

각 염기끼리의 대응에 대해서 값을 주기 위한 스코아 매트릭스로는 FASTA 프로그램(9)의 DNA 스코아 매트릭스(score matrix)를 사용하였다. k 길이의 gap에 대해서는 다음과 같은 gap penalty 함수 gap(k)를 사용하였다.

$$\text{gap}(k)=g+h(k-1),\ k\geq 1$$

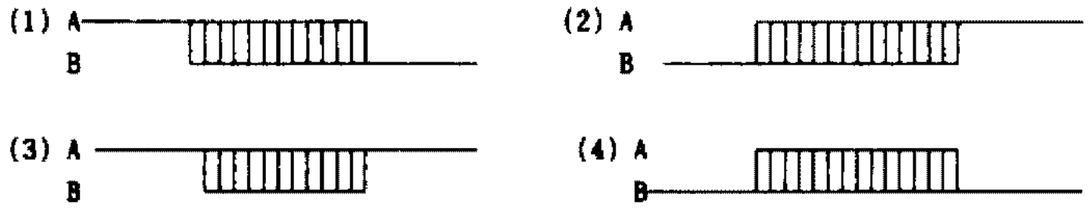

Fig. 2. Four types of overlaps between two fragments A, B.
(1), (2) partial overlaps.
(3), (4) containment of a sequence by another.

위의 식에서 g는 GOP(Gap Open Penalty), h는 GEP (Gap Extension Penalty)이며(10), gap이 한 개일 경우 g 값이 gap penalty가 되며, gap 수가 증가할수록 h 만큼의 값이 gap penalty에 더해진다. 단편들의 정렬시 발생하는 gap은 단편 염기 서열을 결정하는 과정에서 발생된 오류이다. 이 경우 이웃하는 곳에서 동시에 오류가 발생할 확률이 다른 한 곳에서 오류가 발생할 확률보다 작으므로 g 값보다 h 값을 더 크게 설정하였다. 또한, 두 단편의 최대치 정렬시 허용할 수 있는 불일치(mismatch) 염기쌍의 수를 결정하기 위하여, 중첩 유사값(overlap similarity value)을 사용하였다. 이 값은 각 단편의 염기 서열을 알아내는 과정에서 발생하는 오류 확률(r)에 의해서 결정된다. 만약 r 값이 0%에 가까운 경우, 단편 쌍의 중첩이 약 100%의 유사성을 유지해야 한다는 것을 의미한다. 그래서 다음의 식이 가능하다.

$$\text{중첩 유사값} \simeq 1\text{-}2r$$

r 값은 DNA 데이타베이스인 EMBL(European Molecular Biology Laboratory)에서 우연히 오류가 발생할 확률값인 3.55%로 하였다(11).

단편 쌍들의 contig 구성

단편 쌍 정렬 중에서 서로 간의 존재가 모순되어 contig 구성에 반영할 수 없는 경우가 다수 발생하게 된다. 이러한 모순은 입력 단편의 상보 가닥의 고려, contig를 이루게 될 서열의 자체 상동성 존재 등으로부터 발생하게 된다. 상보 가닥에 대해 contig 구성의 모순이 발생할 경우 다음과 같은 두가지 경우로 나누어 처리하였다.

i) 입력 단편과 그의 상보 가닥이 각각의 단편 쌍을 형성하는 경우: 최대치가 높은 단편 쌍의 가닥이 사용되고 낮은 최대치를 가진 단편 쌍은 정렬 목록에서 제거하였다.

ii) 입력 단편과 그의 상보 가닥을 가진 두 단편 쌍이 같은 최대치를 가진 경우: 그 단편 쌍 보다 낮은 최대치를 가진 단편 쌍들 중, 해당 단편을 포함되면서 가장 높은 최대치를 가진 단편 쌍에 의해서 그 반대 가닥을 가진 단편 쌍들이 제거된다.

Contig 구성에서 각 단편 쌍 정렬을 반영하는 기준으로 앞 단계에서 구한 정렬 최대치를 사용하였다. 따라서, contig 구성에서 모순되는 정렬 쌍은 정렬 최대치를 우선으로 하여, 한 단편 쌍 씩을 contig에 포함시킬 것인지 배제할 것인지를 결정한다.

Contig에 관한 정보를 저장하기 위한 데이터 구조로서 연결 목록을 사용하였다. Contig 구성 후, 각 자리의 대표(consensus) 염기는 가장 다수인 염기로 결정하고, 같은 수로 존재하는 염기의 경우 IUB-IUPAC 염기로 나타내었다.

연결 목록에 저장된 각 단편 서열과 대표 서열은 텍스트나 포스트스크립트로 출력할 수 있도록 하였다. 포스트스크립트 양식의 출력에는 포스트스크립트 level 1에서 지원하는 연산자(operator)들을 사용하였다. 좌표 계산, 반복 출력, 문자 크기 변환 등은 C 프로그램에서 수행하여, 최종적으로 출력되는 포스트스크립트 문서는 계산된 수치로 인쇄하는 명령만을 수행하도록 하였다.

결과 및 고찰

본 프로그램의 테스트 데이터를 얻기 위해서, GenBank 데이터베이스에서 크기가 다른 여러 개의 염기 서열을 추출하였다. 단편화 프로그램을 사용하여 각 sequence로부터 임의의 위치에서 200~300 염기 정도 크기의 단편을 만들었으며, 이 경우 만들어진 단편 수는 전체 단편의 염기 수가 original sequence의 약 3배가 되도록 설정하였다(즉, original sequence 길이가 1000인 경우, 생성된 단편의 총 염기 수는 3000개가 된다). 또한 염기 서열을 결정할 경우 발생되는 오류를 고려하여 각 단편에서 2~3개 정도의 염기를 제거하였으며, 입력 단편에 IUB-IUPAC 염기를 나타내기 위해서 2~3개의 임의의 염기를 선정하여 그 염기를 포함하는 모호한 염기로 전환된 IUB-IUPAC 염기를 사용하였다(즉, A->M, T->K, G->V 등으로 전환).

위와 같이 생성된 단편들을 FAP의 입력 양식인 FASTA 양식(9)으로 하나의 화일에 저장한 후, 본 프로그램을 수행하였다. 실행 시간은 단편 수가 많을수록 증가하며, 전체 프로그램 실행 시간의 대부분을 '단편 쌍의 최대치 정렬'을 구하기 위한 단계가 차지하는 것으로 나타났다. '중첩 가능성이 있는 단편 쌍 선별' 단계에서 사용된 시간은 단편의 수에 상관없이 프로그램 실행 시간에 큰 영향을 미치지 않는 것으로 나타났다(Table 1).

본 프로그램의 사용 양식은 '<fap> <input_file> [<output_file>]'로 간단하게 구성하였다. 결과 내용은 화면으로 출력하거나 출력 화일로 저장할 수 있으며, 사용자가 출력 양식, 한 줄당 염기의 수를 지정할 수 있도록 하였다. 결과 내용은 크게 세 부분으로 나누었다(Fig. 3). 첫 번째 부분은 입력 단편과 구성된 contig에 관한 정보를 나타내고, 두 번째 부분은 contig 구성에 관여하는 단편들의 정렬된 상태를 염기 단위로 표현하면서 대표 서열과의 관계를 나타내었으며, 마지막 부분은 구성된 contig들의 대표 서열을 나열하였다. 특히 포스트스크립트 출력의 경우, 마지막 부분에서는 전체적으로 단편 위치와 다른 단편들 사이의 관계를 일목요연하게 볼 수 있도록 모든 단편을 라인으로 나타내었다(Fig. 4).

본 프로그램 실행 속도의 대부분을 차지하는 단계는 각 단편 쌍들끼리의 최대치 정렬을 구하기 위한 정렬 계산 과정이다. 그러므로 이 단계로 들어오는 단편 쌍 수를

Table 1. Performance analysis made with four sample sequences from Genbank database

Sequence name (ACCESSION NO.)	Chimpanzee cytochrome coxidasse (M34599)	Retinoid-binding protein (U18602)	C.aethiops gene (g176503)	UV-damaged DNA-binding protein (L20216)
Sequence size	956	1811	2304	4181
Number of fragments	13	26	31	59
Number of total comparison pairs	312	1300	1860	6844
Total time[a]	26 sec.	83 sec.	113 sec.	447 sec.
Time only for obtaining maximum scores[b]	15 sec.	81 sec.	109 sec.	442 sec.

[a]Total execution time is linear proportional to the number of comparison pairs, that is, it is a quadratic function of the number of fragments. [b]The time of dynamic programming for alignments between all fragments is a major term in the total execution time.

최대한 줄임으로서 실행 속도를 빠르게 할 수 있다. 이를 위해서는 contig 형성에 기여하지 않는 즉 중첩 가능성이 없는 단편 쌍들의 제거를 위한 보다 효율적인 알고리즘 개발이 중요하다.

입력 단편의 염기 수가 1000개 이하로 비교적 적은 경우에는 본 프로그램으로 충분히 실행할 수 있지만, 단편 염기 수가 많은 경우 메모리 부족 현상을 일으킬 수 있다. 이는 본 프로그램에서 단편 쌍의 최대치 정렬을 구하기 위한 데이터 구조(data structure)로 2차원 배열을 사용하기 때문이다. 이러한 문제점을 해결하기 위해서는 최대치 정렬을 linear space를 사용하여 구하는 알고리즘을 채택하여 본 프로그램을 수정할 필요가 있을 것이다 (12). 그러나 contig 형성을 위해서는 단편 그 자체가 메모리에 수용되어야 하므로, linear space 알고리즘을 적용할 경우에도 메모리 한계에 영향을 받는다. 현재 사용되는 대부분의 sequencing machine의 용량을 고려하면 본 프로그램이 실용적이라고 볼 수 있지만 거대 contig를 형성하는 경우와 PC급의 기계에서 프로그램을 실행하는 경우 등을 고려하면 space와 실행 시간에 대한 면이 많이 보완되어야 될 것이다. 한편 서열 편집 등의 부가 기능과 사용자 인터페이스 등이 실용적으로 개선되고, ORF 분석, 상동성 분석 프로그램 및 데이터베이스와의 연결 관계 등도 고려되어야 할 것이다.

본 프로그램의 소스 코드는 전자 우편(bulee@geri4680.geri.re.kr)을 통하여 요청하면 무료로 보내줄 예정이다.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    GENERAL INFORMATION                               +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1) The number of Fragments = [ 4 ]
 2) A maximum size fragment = [ G_2-vgt <300>]
 3) A minimum size fragment = [ (null) <250>]
 4) Average depth = [ 2.48 ]
 5) The number of contigs = [ 1]
 6) The length of CONSENSUS sequence = [ 423 ]
 7) Base composition of CONSENSUS sequence
     A = [ 116 ]     C = [ 82 ]      G = [ 59 ]     T = [ 152 ]     M = [ 4 ]
     R = [ 1 ]       W = [ 6 ]       S = [ 1 ]      Y = [ 1 ]       K = [ 0 ]
     V = [ 0 ]       H = [ 0 ]       D = [ 0 ]      B = [ 1 ]       N = [ 0 ]

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    DETAILED DISPLAY OF CONTIGS                       +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+' means given segment; '-' means reverse complement
                >>>>>>>>>>>>>>> CONTIG [ 1] <<<<<<<<<<<<<
(+)G_1          : TATTT TAGAG ACCCA AGTTT TTGAC CTTTT CCATG TTTAC ATCAA TCCTG TAGST

[CONSENSUS]     : TATTT TAGAG ACCCA AGTTT TTGAC CTTTT CCATG TTTAC ATCAA TCCTG TAGST

(+)G_1          : GATTG GGCAG CCATT TAAGT ATTAT TATAG ACATT TTCAC TATCC CATTA AAACC
(+)G_4bua       :                                                   AATA AATAC

[CONSENSUS]     : GATTG GGCAG CCATT TAAGT ATTAT TATAG ACATT TTCAC TATCC CAWTA AAHMC

(+)G_4bua       : CAAAA AAATA GTATA TCTAC ATAGA ATTTC ACATA AAATA AACT- GTTTT -CTAT
(+)G_1          : CTTTA TGCCC ATACA TCATA ACACT ACTTC -CTAC CCATA AGCTC CTTKT AACTT
(+)G_2-vgt      :             ATACA TCATA ACACT ACTTC -CTAC CCATA AGCTC CTTTT HACTT
(+)G_3345       :                                     CATA AGCTC CTTTT AACTT

[CONSENSUS]     : CWMWA WRMYM ATACA TCATA ACACT ACTTC ACTAC CCATA AGCTC CTTTT AACTT

(+)G_4bua       : GTGAA AATKA ACCTA AAAAT --ATG CTTTG CTTAT GTTT- AAGAT GTCAT GCTTT
(+)G_1          : GTTAA AGTCT TGCTT GAATT A-AAG ACTTG TTTAA ACAC- AAAAT -TTAG ACTTT
(+)G_3345       : GTTAA AGTCT TGCTT GAATT A-AAG ACTTG TTTAA ACAC- AAAAT -TTAG ACTTN
(+)G_2-vgt      : GTTAA AGTCT TGCTT GAATT ATAAG ACTTG TTTAA ACACA AAAAT -TTAG AGTTT

[CONSENSUS]     : GTTAA AGTCT TGCTT GAATT ATAAG ACTTG TTTAA ACACA AAAAT GTTAG ACTTT

(+)G_4bua       : TTATC AGTTG AGGAG TTCAG CTTAA TAATC CTCTA CGATC TTAAA CAAAT AGGAA
(+)G_1          : TACTC AACAA AAGTG ATTGA TTGVT TGATT GATT
(+)G_3345       : TACTC AACAA AAGTG ATTGA TTGAT TGATT GATTG ATTGA TGGTT TACAS TAGGA
(+)G_2-vgt      : TA-TC AACAA AAGTG ATTGA TTGAT TGATT GATTG MTTGA TGGTT TACAG TAGGA

[CONSENSUS]     : TACTC AACAA AAGTG ATTGA TTGAT TGATT GATTG MTTGA TGGTT TACAB TAGGA

                      ----- omitted -----

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    CONSENSUS SEQUENCE                                +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
=> CONTIG [ 1] :
TATTTTAGAGACCCAAGTTTTTGACCTTTTCCATGTTTACATCAATCCTGTAGSTGATTGGGCAGCCATTTAAGTATTAT
TATAGACATTTTCACTATCCCAWTAAAWMCCWWWAWRMYMATACATCATAACACTACTTCACTACCCATAAGCTCCTTTT
AACTTGTTAAAGTCTTGCTTGAATTATAAGACTTGTTTAAACACAAAAATGTTAGACTTTTACTCAACAAAGTGATTGA
TTGATTGATTGATTGMTTGATGGTTTACABTAGGACTTCATTCTAGTCATTATAGCTGCTGGCAGTATAACTGGCCAGCC
TTTAATACATTGCTGCTTAGAGTCAAAGCATGTACTTAGAGTTGGTATGATTTATCTTTTTGGTCTTCTATAGCCTCCTT
CCCCATCCCCATCAGTCTTAATC
```

Fig. 3. An output of the fragment assembly program as a text format.

An output consists of three parts : genenal information on fragments and contigs, detailed structure of contigs, and contig sequences.

요 약

본 연구에서는 매우 긴 DNA 염기 서열을 밝혀내려는 경우 발생하는 contig 구성 문제를 해결하기 위한 알고리즘을 구성하고 응용 프로그램을 개발하였다. 프로그램의 입력은 일정한 방향성을 가지지 않은 DNA 단편들의 IUB-IUPAC 염기 서열이며, 출력은 구성된 contig 염기 서열로서 텍스트나 포스트스크립트 양식으로 되어 있다.

본 프로그램의 알고리즘은 3 단계로 구성하였다. 첫 번째 단계는, 모든 가능한 단편 쌍들 중에서 중첩 가능성이 있는 단편 쌍들을 선별하는 단계로, 단편 쌍의 최소 중첩 길이에 의해서 결정된다. 두 번째 단계는 선별된 단편 쌍들의 중첩 정도를 측정하는 단계이다. 마지막 단계은 각 단편 쌍들을 중첩 정도를 기준으로 정렬 최대치 값의 순으로 정리한 후, 중첩 정도가 큰 단편 쌍부터 contig를 구성하여 그 결과를 연결 목록에 저장하였다. 연결 목록에 저장된 결과는 화면으로 보여주면서 동시에 텍스트 혹은 포스트스크립트 양식으로 파일로 저장된다.

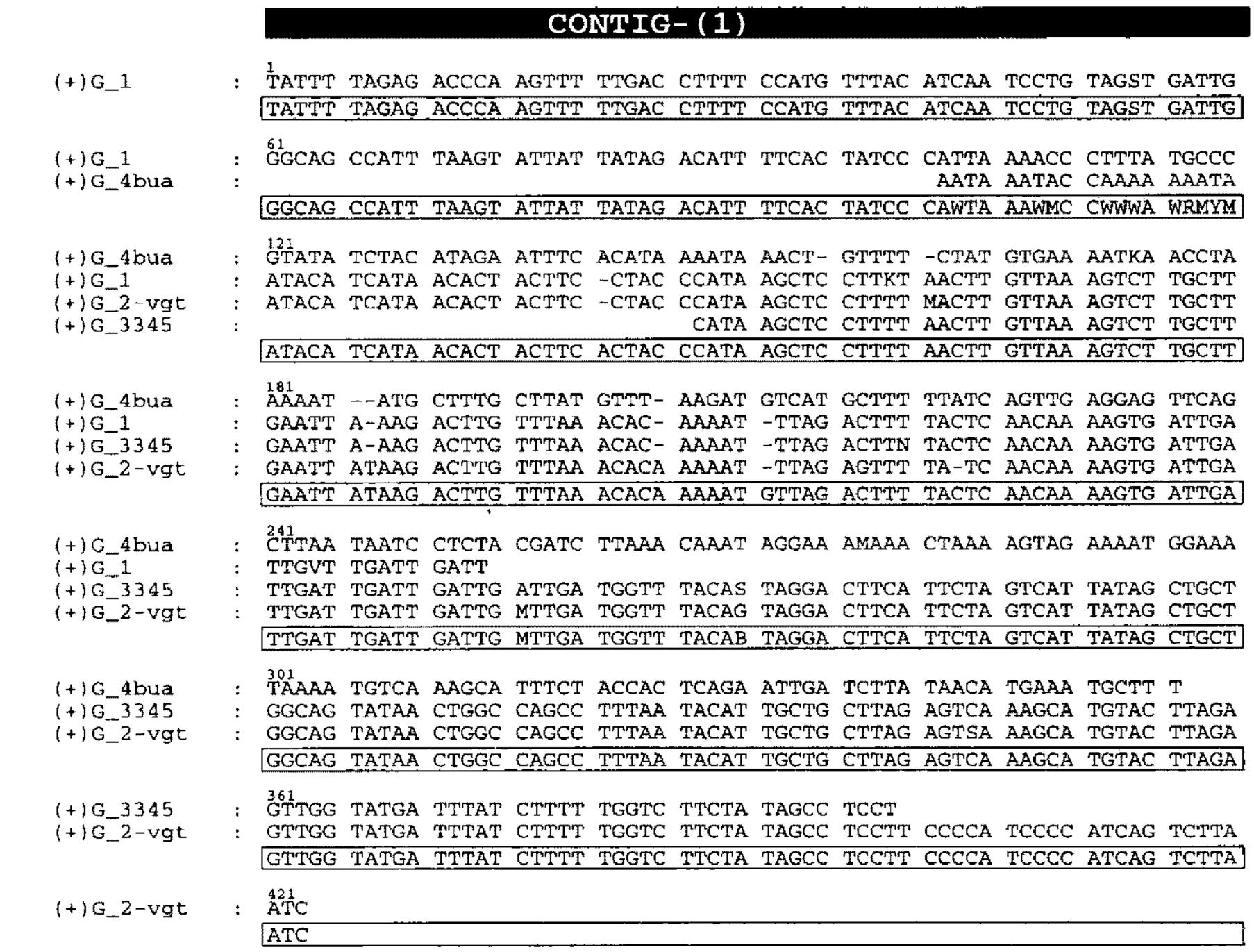

Fig. 4. An output of fragment assembly program as a PostScript format.
In detailed display of contigs, consensus bases are in gray rectangle, and in linear display of contig sequences, line means fragment sequence.

본 프로그램의 테스트 데이타로서 GenBank 데이터베이스에서 일정 길이의 서열을 추출하여 적당한 크기의 단편으로 자른 후, 일부 염기를 변이 및 IUB-IUPAC 염기로 전환시킨 단편 서열들을 사용하였고, 프로그램 단계별 및 단편 서열 수에 따라 실행 시간을 분석하였다.

참고문헌

1. Staden, R. 1980. A new computer method for the storage and manipulation of DNA gel reading data. *Nucl. Acids Res.* **8**: 3673-3694.
2. Hannu, P., H. Soderlund and E. Ukkonen. 1984. SEQAID: a DNA sequence assembling program based on a mathematical model. *Nucl. Acids Res.* **12**: 307-321.
3. Xiaoqiu, H. 1992. A Contig Assembly Program Based on Sensitive Detection of Fragment Overlaps. *Genomics* **14**: 18-25.
4. Cornish, B. 1985. A Nomenclature for incompletely specified bases in nucleic acid sequences:recommendation. *Nucl. Acids Res.* **16**: 3021-3030.
5. Henry, M. and M. Campione. 1992. *PostScript by exam-*

ple, Pp 2-346. 1th ed. Addison Wesley, Massachusetts.
6. Eric, S. L. and M. S. Waterman. 1988. Genome Mapping by Fingerprinting Random Clones: A Mathematical Analysis. *Genomics* **2**: 231-239.
7. Sedgewick, R. 1988. *Algorithm*, Pp 231-245. 2th ed. Addison Wesley, Massachusetts.
8. Smith, T. F. and M. S. Waterman. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* **147**: 195-197.
9. Peason, W. R. and D. J. Lipman. 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA.* **85**: 2444-2449.
10. Spouge, J. L. 1991. Fast optimal alignment. *CABIOS* **7**: 1-7.
11. Kristensen, T., R. Lopez and H. Prydz. 1992. An estimate of the sequencing error frequency in the DNA sequence database. *DNA seq.* **12**: 343-346.
12. Hirschberg, D. S. 1975. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Communications of the ACM* **18**: 341-343.

(Received 22 April 1997)