Séquençage : Assemblage de Génomes...

Encadrant : Mr A. El Hassouny elhassounyphd@gmail.com

Zakarya El Khiyati Yassine Birouk

Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes

Presentation du projet, 2017

Plan

1. Assemblage de genome
2. Shortest superstring
   - Graphe des chevauchements
   - Algorithme glouton
   - Shorstest superstring est difficile
3. chercher un motif de chaîne spécifique
4. Implementation en C : Fonctions principales
   - int overlap(char* a, char* b, int min_lenth)
   - void concatenateOverlap(char* str1,char* str2, int k)
   - pick pick_maximal_overlap(char *reads[], unsigned, int k)
   - void greedy(char* reads[], unsigned size, int k)
   - int recherche(char *f_souche,char *orig)

Assemblage

Principe

Assemblage

Donnee : un ensemble de lectures prises dans le genome cible
Sortie : la sequence du genome cible

Idee

utiliser les chevauchements entre lectures pour savoir comment les aggreger et leur ordre relatif

modelisation informatique

Shortest Common Superstring

Shortest common superstring

Exemple :

```
ACCC          ACTG
ACTG            CTGA
CCCT              TGAC
CCTA                GACC
CTAG                  ACCC
CTGA                    CCCT
GACC                      CCTA
TAGT                        CTAG
TGAC                          TAGT

                ACTGACCCTAGT
```

Shortest common superstring

Trois Approches majeurs

- Gloutonne
- Graphe des chevauchements
- Graphe de de Burijin

Outline

Shortest superstring

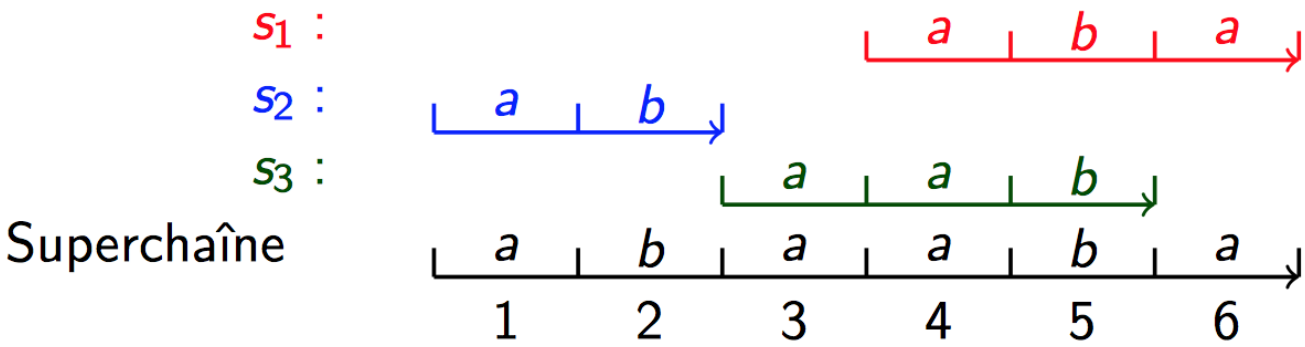
Modelisation simple de l'assemblage

spurstring

Soient $S := s_1, ..., s_n$ un ensemble de$n$ mots. Le mot w est une superchaine de S ssi si est une sous-chaine de w.

Shortest common superstring

Trouver une superchaine de longueur minimale.

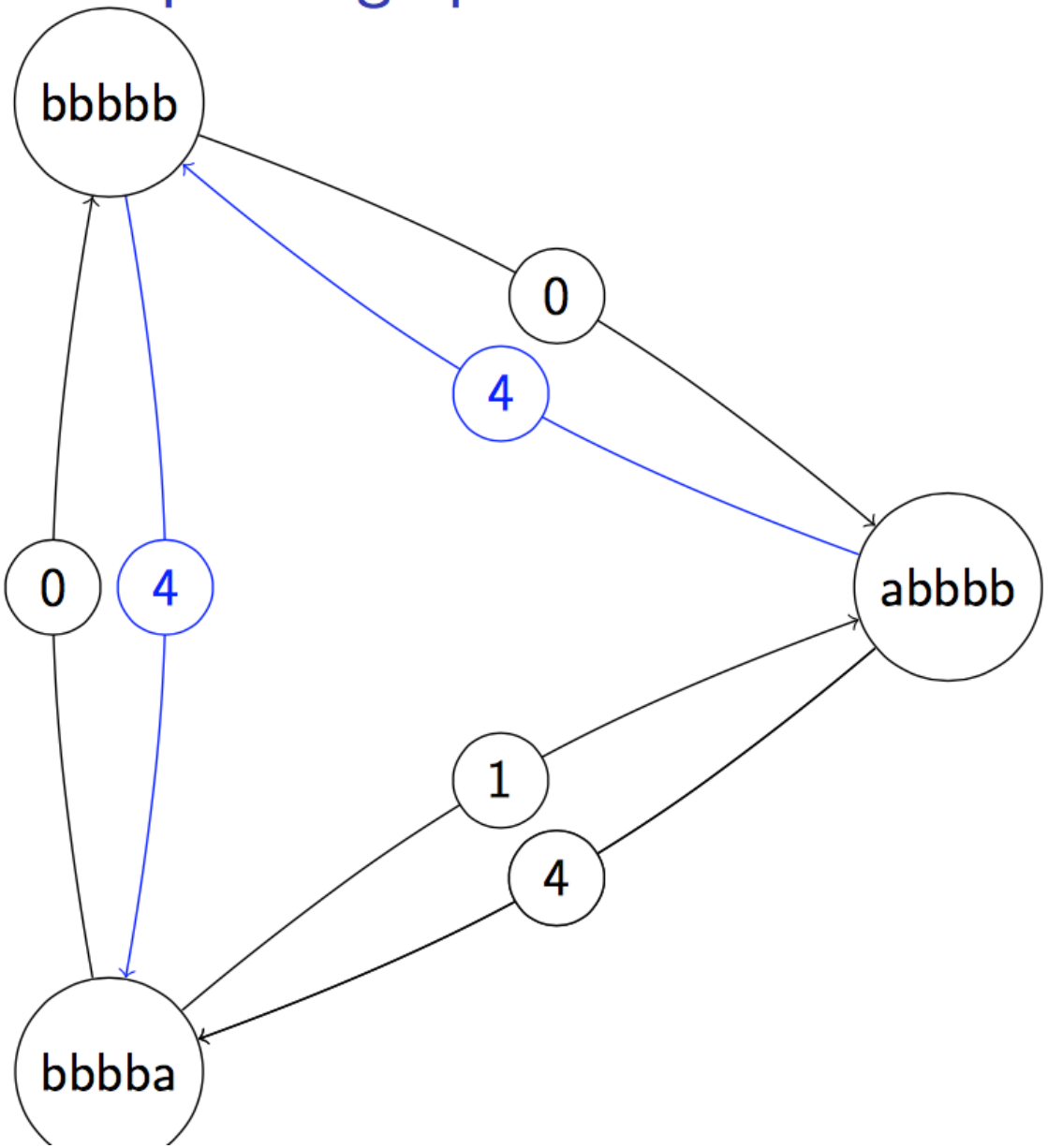


Graphe de chevauchements

Graphe oriente pondere tel que :

- un noeud par mot
- une arete relie deux mots s'ils ont un chevauchement non nul
- poids d'une arete : longueur de leur chevauchement maximal

Exemple de graphe de chevauchements



$S :=$
$\{abbbb, bbbbb, bbbba\}$

Outline

Algorithme glouton

Principe

Les algorithmes gloutons appliquent une opération de base : compte tenu de toute lecture ou contig, ajouter un contig. Cette opération de base est répétée jusqu'à ce qu'il n'y plus d'opérations possibles. Chaque opération utilise le score maximal de chevauchement afin de faire le prochain recollement.

Equivalent a l'algorithme glouton sur le overlap graph

Outline

Algorithme glouton

Théorème

- *NP-difficile meme si les reads sont de longueur de 3*
- *conjecture : algorithme glouton approxime la longueur de la SCS a un ratio 2*

interface utilisateur

L'interface utilisateur

```
yhn-HP-EliteBook-8540p:~/Bureau/ensias/projet/proj$ ./walo
==================================================menu:=====================================

                                    1-ASSEMBLAGE DE SEQUENCES
                                    2-RECHERCHE DES FAUSSES SOUCHES

   entrer le numero de la fonction que vous voullez utiliser ou ESC pour quitter:

=============================================================================================
```

Outline

Fonctions principales

la fonction 'overlap' permet de donner le nombre maximal de chevauchement entre deux séquences.

overlap

```
int overlap(char* a, char* b, int min_lenth){
    int start = 0;
    char* _a;// servira comme sous chaine de a
    char* _b;// servira comme sous chaine de b
    _b =(char*) malloc((min_lenth + 1));
    substring(b, _b, 0, min_lenth);
    while(1){
        _a =(char*) malloc(strlen(a) - start + 1);
        _a =(char*) realloc(_a,(strlen(a) - start) + 1);
        substring(a, _a, start, strlen(a));
        start = find_str(_a, _b) == -1 ? -1 : find_str(_a, _b) + start;
        if( start == -1 ){
            free(_a);
            free(_b);
            return 0;}
        _a =(char*) realloc(_a, (strlen(a) - start) + 1);
        substring(a, _a, start, strlen(a));

        if( startswith(b, _a) )0{
            free(_a);
            free(_b);
            return strlen(a) - start;}
        start++;
        free(_a);}
}
```

example

```
    enrer la 1 eme sequence:
CGTATGGTCTGACTGTAGCGT
    enrer la 2 eme sequence:
ACTGTAGCGTATGCTGAACTG

le nombre de chevauchement entre str1 et str2 est: 10
```

Outline

Fonctions principales

après la detection des deux chaines qui se chevauchent, cette fonction les concatinent.

concatenateOverlap

```
void concatenateOverlap(char* str1,char* str2, int k)
{

    unsigned long len  = strlen(str1);
    unsigned long len2 = strlen(str2);
    memcpy(str1 + len, str2 + k, len2 - k + 1);

}
```

example

```
    enrer la 1 eme sequence:
CGTATGGTCTGACTGTAGCGT
    enrer la 2 eme sequence:
ACTGTAGCGTATGCTGAACTG

le resultat est: CGTATGGTCTGACTGTAGCGTATGCTGAACTG
```

Outline

Fonctions principales

Calcule le score de chevauchement pour chaque paire(ordone) de séquences et retourne celui pour lequel ce score est maximale.

pick_maximal_overlap

```
pick pick_maximal_overlap(char** reads, unsigned size, int k)
{
    int a = 0; int b = 1;
    int olen;
    int best_olen = 0;
    for (int i = 0; i < size-1; ++i)
    {
        for (int j = i + 1; j < size; ++j)
        {
            olen = overlap(reads[i], reads[j], k);
            if (olen > best_olen)
            {
                a = i;
                b = j;
                best_olen = olen;

            }
            olen = overlap(reads[j], reads[i], k);
            if (olen > best_olen)
            {
                a = j;
                b = i;
                best_olen = olen;
            }
        }
    }
    struct pick best_pick = {a, b, best_olen};
    return best_pick;
}
```

example

```
    enrer la 1 eme sequence:
CGTATGGTCTGACTGTAGCGT
    enrer la 2 eme sequence:
ACTGTAGCGTATGCTGAACTG
    enrer la 3 eme sequence:
CGTATGCTGAACTGTAAGGTA

les chaines qui ont la max de chevauchement sont:
 ACTGTAGCGTATGCTGAACTG et CGTATGCTGAACTGTAAGGTA
 avec 14 chevauchement
```

Outline

Fonctions principales

Colle les deux séquences ayant un score maximal en une nouvelle séquence et reitere ce processus avec le nouvel ensemble de séquences (qui contient maintenant une séquence en moins) jusqu'à ce qu'il ne reste qu'une séquence, qu'elle retourne.

greedy

```
void greedy(char* reads[], unsigned size, int k)
{
    pick best = pick_maximal_overlap(reads, size, 1);
    int i = 0;
    while(size - i > 1)
    {
        reads[best.a] = realloc(reads[best.a], strlen(reads[best.b]) + strlen(reads[best.a]) - best.key + 1);

        concatenateOverlap(reads[best.a], reads[best.b], best.key);

        _remove(&reads, size - i, best.b);
        i++;
        best = pick_maximal_overlap(reads, size - i, 1);
      }
    printf("\n\n\n\n%s\n\n\n\n", reads[0]);
    }
```

example

```
1
    enrer la 1 eme sequence:
CTCATCG
    enrer la 2 eme sequence:
TCGATGC
    enrer la 3 eme sequence:
ATGCCGTAC


    le résultat est:
CTCATCGATGCCGTAC
```

Outline

'rechercher' est la fonction qui cherche une fausse souche dans une séquence qui lui est donnee par l'utilisateur, en s'apuyant sur la fonction find$_str$.

```
int recherche(char *f_souche,char *orig){
    int test1;
    test1=find_str(orig,f_souche);
    return test1;

}
```

example

```
2
        donner la chaine originale:
                ^^^^^^^^^^^^^^^^
CTCAGCTAGGTCCAATGCCGTGGAACTGGTGGAACCC

        donner la fausse souche:
CCAATGCCGTGGAA

la sequence contient cette fausse souche
```

Bibliographie I

E. Ukkonen, J. Tarhio
A Greedy Approximation Algorithm for Constructing Shortest Common Superstrings.
Theoretical Computer Science, LNBI vol. 57, p. 131-145, 2014.

B. Cazaux, T. Lecroq, E. Rivals
Approximation of greedy algorithms for Max-ATSP, Maximal Compression, Maximal Cycle Cover, and Shortest Cyclic Cover of Stringss.
8486, pp. 89-99, Springer Verlag, 2014.