Restriction Mapping Algorithms

- In this presentation, we will give algorithms to reconstruct the ordering of segments produced from an unknown DNA sequence by using restriction enzymes. We will consider only partial restriction digest.
- These algorithms are not used in modern day biotechnology but the techniques illustrate the application of branch-and-bound techniques in mathematical biology and was a hot research topic in the 70s and 80s.

Discovering Restriction Enzymes

- *Hin*dII - first restriction enzyme – was discovered accidentally in 1970 while studying how the bacterium *Haemophilus influenzae* takes up DNA from the virus
- Recognizes and cuts DNA at sequences:
  - GTGCAC
  - GTTAAC

Discovering Restriction Enzymes

Werner Arber **Daniel Nathans** **Hamilton Smith**

Werner Arber – discovered restriction enzymes

Daniel Nathans - pioneered the application of restriction for the construction of genetic maps

Hamilton Smith - showed that restriction enzyme cuts DNA in the middle of a specific sequence

"My father has discovered a servant who serves as a pair of scissors. If a foreign king invades a bacterium, this servant can cut him in small fragments, but he does not do any harm to his own king. Clever people use the servant with the scissors to find out the secrets of the kings. For this reason my father received the Nobel Prize for the discovery of the servant with the scissors".

Daniel Nathans' daughter
(from Nobel lecture)

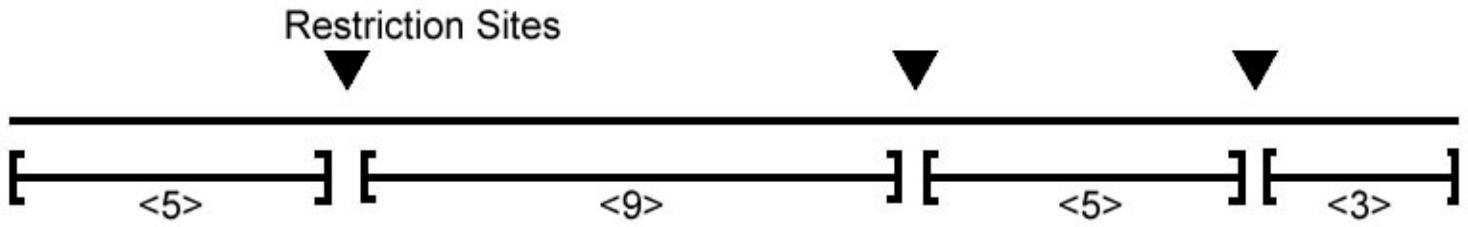
Recognition Sites of Restriction Enzymes

Enzyme	Source Microorganism	Recognition Site*	Ends Produced
BamHI	*Bacillus amyloliquefaciens*	↓ -G-G-A-T-C-C- -C-C-T-A-G-G- ↑	Sticky
*Eco*RI	*Escherichia coli*	↓ G A A T T C C T T A A G ↑	Sticky
*Hin*dIII	*Haemophilus influenzae*	↓ -A-A-G-C-T-T- -T-T-C-G-A-A- ↑	Sticky
*Kps*I	*Klebsiella pneumonia*	↓ -G-G-T-A-C-C- -C-C-A-T-G-G- ↑	Sticky

Molecular Cell Biology, 4th edition

Full Restriction Digest

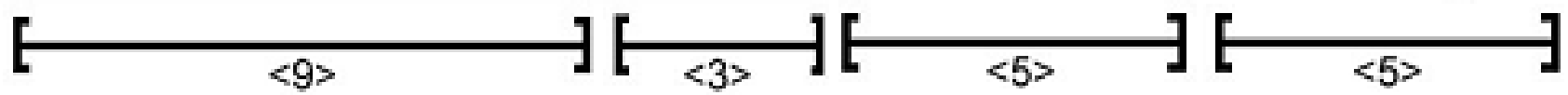
- Cutting DNA at each restriction site creates multiple restriction fragments:



- Is it possible to reconstruct the order of the fragments from the sizes of the fragments {3,5,5,9} ?

Full Restriction Digest: Multiple Solutions

- Alternative ordering of restriction fragments:



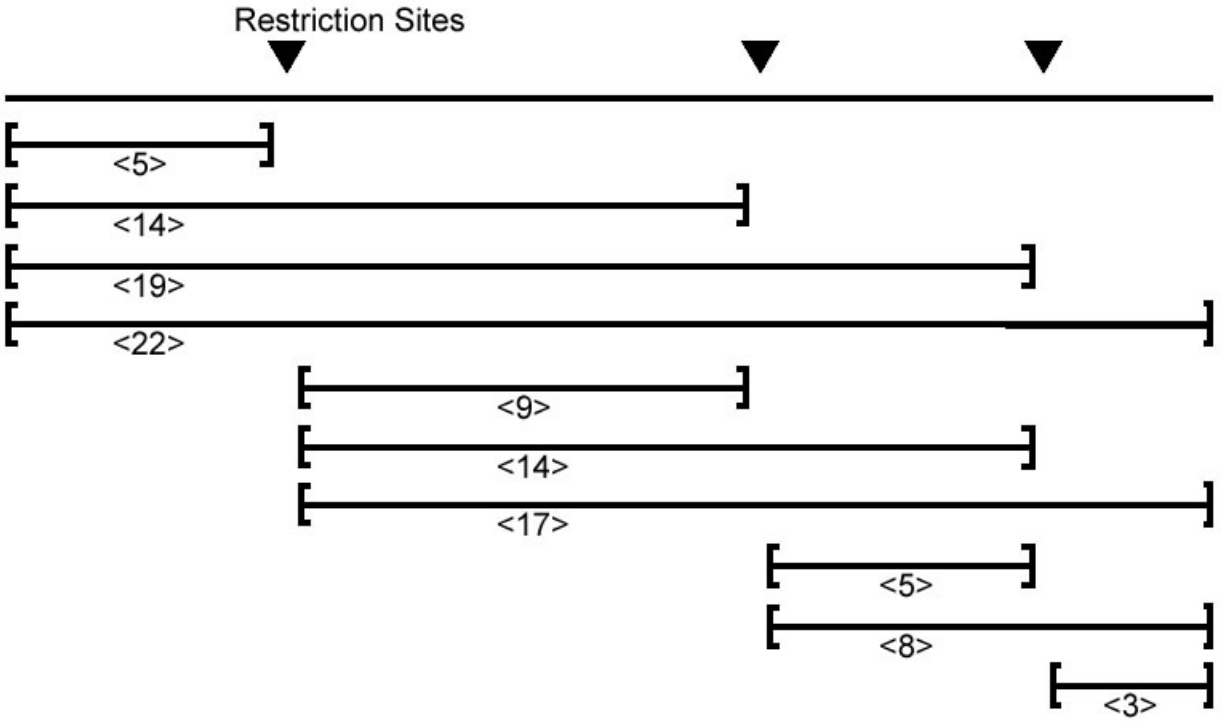
vs

Partial Restriction Digest

- The sample of DNA is exposed to the restriction enzyme for only a limited amount of time to prevent it from being cut at all restriction sites
- This experiment generates the set of all possible restriction fragments between every two (not necessarily consecutive) cuts
- This set of fragment sizes is used to determine the positions of the restriction sites in the DNA sequence

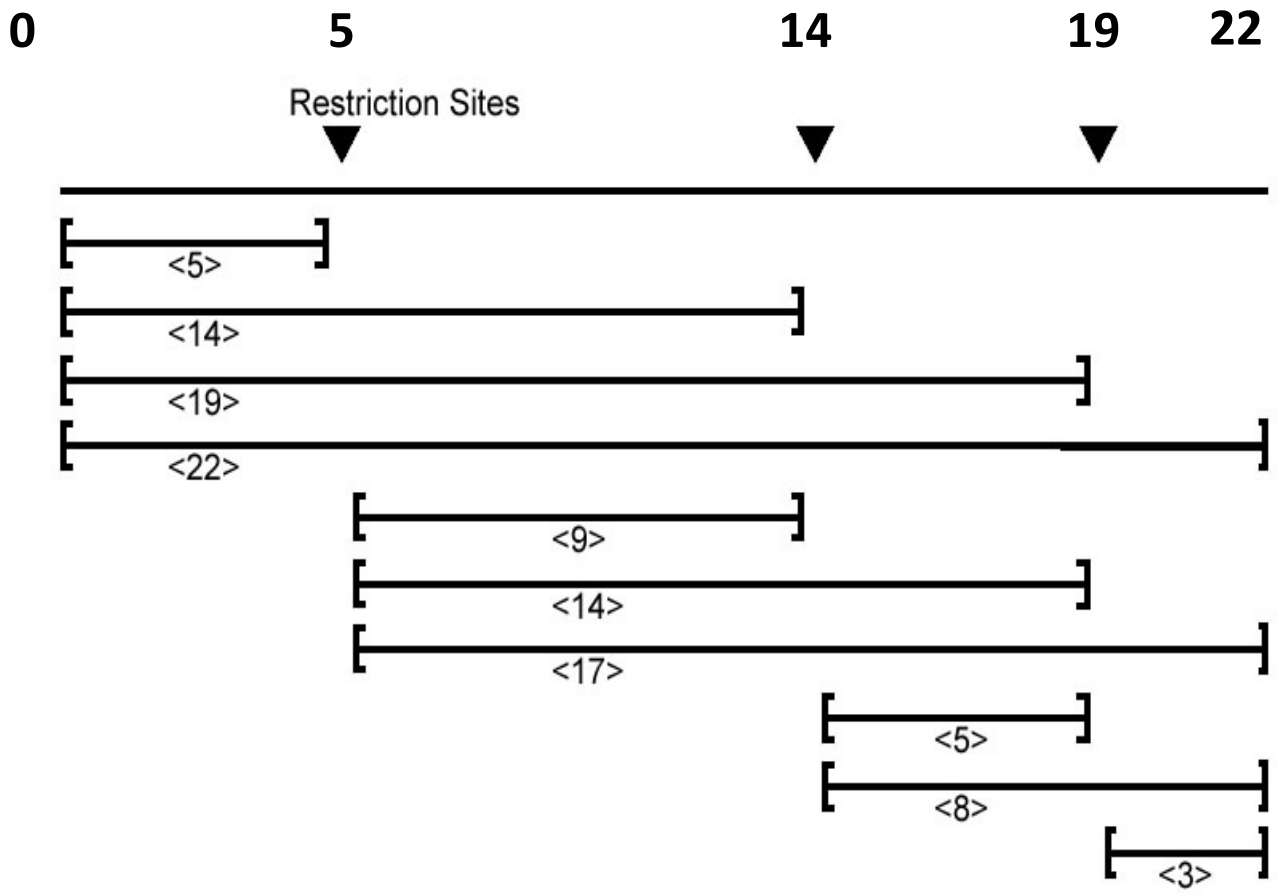
Partial Digest Example

- Partial Digest results in the following 10 restriction fragments:



Multiset of Restriction Fragments

We assume multiplicity of a fragment can be detected, i.e., the number of restriction fragments of the same length can be determined (e.g., by observing twice as much fluorescence intensity for a double fragment than for a single fragment)



Multiset: {3, 5, 5, 8, 9, 14, 14, 17, 19, 22}

Partial Digest Fundamentals

X: the set of ***n*** integers representing the location of all cuts in the restriction map, including the start and end (0,5,14.19,22)

n: the total number of cuts (=5)

D*X*: the multiset of integers representing lengths of each of the ${}^{n}C_{2}$ fragments produced from a partial digest (multiset). This set is the mutual difference set of all elements in *X*

One More Partial Digest Example

X	0	2	4	7	10
0		2	4	7	10
2			2	5	8
4				3	6
7					3
10					

Representation of D$\boldsymbol{X}$ = {2, 2, 3, 3, 4, 5, 6, 7, 8, 10} as a two dimensional table, with elements of

$$\boldsymbol{X} = \{0, 2, 4, 7, 10\}$$

along both the top and left side. The elements at (i, j) in the table is $x_j - x_i$ for $1 \leq i < j \leq n$.

Partial Digest Problem: Formulation

Goal*:* Given all pairwise distances between points on a line, reconstruct the positions of those points

- Input: The multiset of pairwise distances *L*, containing ${}^{n}C_2 = n(n-1)/2$ integers
- Output: A set *X*, of *n* integers, such that D*X* = *L*
- *L* is given find *X.*

Partial Digest: Multiple Solutions

- It is not always possible to uniquely reconstruct a set ***X*** based only on *L*=D***X***. The solutions are not unique.

- For example, the set

$$X = \{0, 2, 5\}$$

and

$$(X + 10) = \{10, 12, 15\}$$

both produce D***X***={2, 3, 5} as their partial digest set.

- The sets {0,1,2,5,7,9,12} and {0,1,5,7,8,10,12} present a less trivial example of non-uniqueness. They both digest into:

{1, 1, 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9, 10, 11, 12}

Homometric Sets(Two sets **A** and **B** such that D***A***=D***B***)

	0	1	2	5	7	9	12
0		1	2	5	7	9	12
1			1	4	6	8	11
2				3	5	7	10
5					2	4	7
7						2	5
9							3
12							

	0	1	5	7	8	10	12
0		1	5	7	8	10	12
1			4	6	7	9	11
5				2	3	5	7
7					1	3	5
8						2	4
10							2
12							

Brute Force Algorithms

- Also known as exhaustive search algorithms; examine every possible variant to find a solution

- Efficient in rare cases; usually impractical

Partial Digest: Brute Force

1. Find the restriction fragment of maximum length ***M***. ***M*** is the length of the DNA sequence.

2. For every possible set

$$\boldsymbol{X}=\{0, x_2, \ldots, x_{n-1}, M\}$$

compute the corresponding D***X***

5. If D***X*** is equal to the experimental partial digest ***L***, then ***X*** is the correct restriction map

BruteForcePDP

1. BruteForcePDP(*L*, *n*):
2. M <- maximum element in L
3. for every set of $n - 2$ integers $0 < x_2 < \ldots x_{n-1} < M$
4. X <- $\{0, x_2, \ldots, x_{n-1}, M\}$
5. Form DX from X
6. if D$X = L$
7. return X
8. output “no solution”

Efficiency of BruteForcePDP

- BruteForcePDP takes $O(\boldsymbol{M}^{n-2})$ time since it must examine all possible sets of positions.

- One way to improve the algorithm is to limit the values of x_i to only those values which occur in $\boldsymbol{L}$.

AnotherBruteForcePDP

1. AnotherBrutForcePDP(*L*, *n*)
2. *M* <- maximum element in *L*
3. for every set of *n* – 2 integers $0 < x_2 < \ldots x_{n-1} < M$ *from L*
4. *X* <- { $0, x_2, \ldots, x_{n-1}, M$ }
5. Form D*X* from *X*
6. if D*X* = *L*
7. return *X*
8. output “no solution”

The algorithm examines $^{|L|}C_{n-2}$ different sets of integers

but |L|=(n(n-1))/2 so the complexity is still $O(n^{2n-4})$

Efficiency of AnotherBruteForcePDP

- It's more efficient, but still slow
- If $\boldsymbol{L}$ = {2, 998, 1000} ($\boldsymbol{n}$ = 3, $\boldsymbol{M}$ = 1000), BruteForcePDP will be extremely slow, but AnotherBruteForcePDP will be quite fast
- Fewer sets are examined, but runtime is still exponential: $O(\boldsymbol{n}^{2n-4})$.

Branch and Bound Algorithm for PDP

1. Begin with ***X*** = {0}
2. Remove the largest element in ***L*** and place it in ***X***
3. See if the element *fits* on the right or left side of the restriction map
4. When it fits, find the other lengths it creates and remove those from ***L***
5. Go back to step 1 until ***L*** is empty

Branch and Bound Algorithm for PDP

1. Begin with ***X*** = {0}
2. Remove the largest element in ***L*** and place it in ***X***
3. See if the element *fits* on the right or left side of the restriction map
4. When it fits, find the other lengths it creates and remove those from ***L***
5. Go back to step 1 until ***L*** is empty

WRONG ALGORITHM
(may have to backtrack because of the choice at step)

An Example to illustrate the idea for a better algorithm

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0 }

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0 }

Remove 10 from ***L*** and insert it into ***X***. We know this must be the length of the DNA sequence because it is the largest fragment.

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 10 }

We know now n=5 since ${}^{n}C_{2}$=10. So, we begin by setting $\boldsymbol{x_5}$ = 10 so that X = { 0, 10 } and remove 10 from L. The new L is L = { 2, 2, 3, 3, 4, 5, 6, 7, 8}.

An Example

Take next largest element 8 from ***L.*** We have two choices $x_2 = 2$ or $x_4 = 8$. But since the two cases are symmetric, we can assume $x_2 = 2$. We remove elements $x_5 - x_2 = 8$ and $x_2 - x_1 = 2$ from *L. The new sets are:*

$$X=\{0,2,10\} \quad \textit{and} \quad L=\{2,3,3,4,5,6,7\}$$

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7}
X = { 0, 2, 10 }

We have two choices again. We could take 7 from ***L*** and make $\boldsymbol{x_4}$ = 7 or $\boldsymbol{x_3}$ = 3. If we choose $\boldsymbol{x_3}$ = 3 then D($\boldsymbol{x_3}$, X)=(3,1,7)*
But, since 1 is not an element of *L, this is a wrong choice.*
So, we choose explore $\boldsymbol{x_4}$ = 7 and D($\boldsymbol{x_4}$, X) = {7, 5, 3}.
D($\boldsymbol{x_4}$, X) = {7, 5, 3} = {7 – 0, 7 – 2, 10 – 7}

*We define D(***y,** X)= multiset of distances between a point ***y*** and all points in a set ***X***

An Example

The rivised L and X are:

L = {2, 3, 4, 6, }

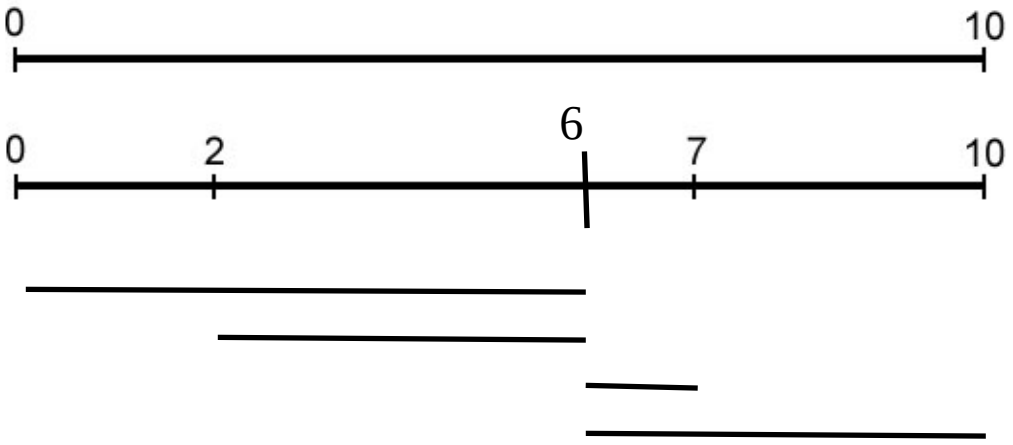
X = { 0, 2, 7, 10 }

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

(In practice, we need to keep the deleted elements at all stages for the purpose of backtracking. They are marked red.)

Take 6 from ***L*** and make $\boldsymbol{x_3} = 6$. Unfortunately
$D(\boldsymbol{x_3}, X) = \{6, 4, 1, 4\}$, which is not a subset of ***L***. Therefore we won't explore.



An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

We are left with one choice $\boldsymbol{x_3} = 4$. D$(\boldsymbol{x_3}, X) = \{4, 2, 3, 6\}$, which is a subset of ***L*** so we will explore this branch. We remove {4, 2, 3 ,6} from ***L*** and add 4 to ***X***.

An Example

An Example

$L = \{\ \}$
$X = \{\ 0, 2, 4, 7, 10\ \}$
L is now empty, so we have a solution, which is X.

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

To find other solutions, we backtrack.

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 10 }

More backtrack.

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 10 }

This time we will explore y = 3. D(y, X) = {3, 1, 7}, which is not a subset of L, so we won't explore this branch.

An Example

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 10 }

We backtracked back to the root. Therefore we have found all the solutions.

Defining D(y, X)

- Before describing PartialDigest, first define

  D(***y***, ***X***)

  as the multiset of all distances between point ***y*** and all other points in the set ***X***

D(y, X) = $\{|y - x_1|, |y - x_2|, \ldots, |y - x_n|\}$ = distances between point y and all points in $X = \{x_1, x_2, \ldots, x_n\}$

Example: D(2,(1,3,4,5))= (1,1,2,3)

PartialDigest Algorithm

PartialDigest(*L*):

width <- Maximum element in *L*
DELETE(*width*, *L*)
X <- {0, *width*}
PLACE(*L*, *X*)

DELETE(*width,L*) simply removes the element '*width*' from L.
PLACE (*L,X*) is a recursive subroutine explained in the next slide.

Partial Digest Algorithm (cont'd)

1. PLACE(*L*, *X*)
2. **if** *L* is empty
3. output *X*
4. return **end if**
5. *y* <- maximum element in *L*
6. Delete*(y, L)* /* *Delete y from L* /*
7. **if** D(*y*, *X*)is a subset of *L*
8. Add *y* to *X* and remove lengths D(*y*, *X*) from *L*
9. PLACE(*L*,*X*) /* Recursive call/*
10. Remove *y* from *X* and add lengths D(*y*, *X*) to *L* ***end if***
11. **if** D(*width*-*y*, *X*) is a subset of *L*
12. Add *width*-*y* to *X* and remove lengths D(*width*-*y*, *X*) from *L*
13. PLACE(*L*,*X*) /* Recursive call/*
14. Remove *width*-*y* from *X* and add lengths D(*width*-*y*, *X*) to *L* ***end if***
15. return

PDP Example

- Given L=(2,2,3,3,4,5,6,7,8,10), *n=5* since $|L|=10$ *and* $^5C_2=10$; *let* w=width.
- w:=10, L=(2,2,3,3,4,5,6,7,8), *X*=(0,10)
- 1. PLACE(*L.X*): *L* not empty; *y*:=8; D(8,(0,10))=(8,2) is a subset of *L*
- *X*=(0,10,8) and *L:=L* - *X*=(2,3,3,4,5,6,7)
- 2. PLACE(*L,X*)): *L* not empty; *y*:=7; D(7,(0,10,8))=(7,3,1) is not in *L*
- D(w-7,(0,10,8))=D(3,(0,10,8))=(3,5,7) is in L
- *X*=(0,10,8,3) and *L:=L* - *X*=(2,3,4,6)
- 3. PLACE(*L,X*): *L* not empty; *y*:=6; D(6,(0,10,8,3))=(6,4,2,3)
- is a subset of *L, and X*=(0,10,8,3,6) and *L is empty*
- *4.* PLACE(*L,X*): *L* empty; output *X*=(0,10,8,3,6) Return
- Remove *y* from *X* and add D(*y,X*) to *L*.

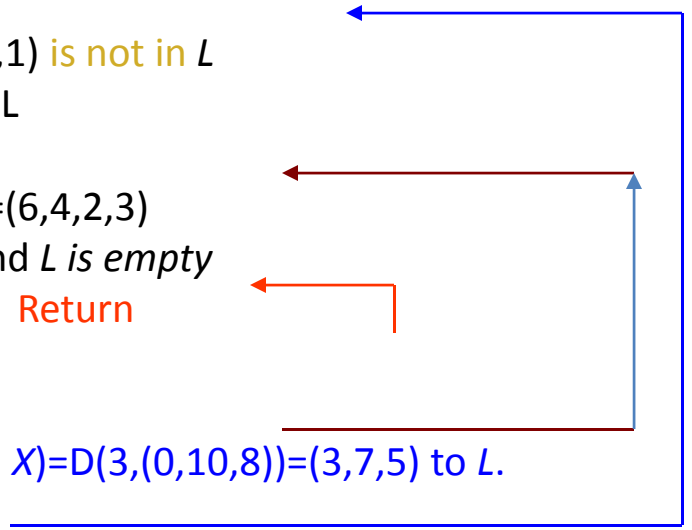
X= (0,10,8,3,), D(6,(0,10,8,3))=(6,4,2,3) and L =(6,4,2,3)

- Remove *w-y=10-7=3* from *X* (=0,10,8,3) and add D(*w-y, X*)=D(3,(0,10,8))=(3,7,5) to *L*.

yielding X=(0,10,8) and L=(2,3,3,4,5,6,7)

- Remove *y* =8 from *X* (0,10) and add D(*y, X*)= D(8,(0,10))=(6,2) to *L*.

This gives back L=(2,2,3,3,4,5,6,7,8), *X*=(0,10) and *w* is still 10. We can start over and take a different possible choice for *y=2 and get another solution for X*=(0,2,4,7,10).



Analyzing PartialDigest Algorithm

- Still exponential in worst case, but is very fast on average
- Informally, let $T(\mathbf{n})$ be time PartialDigest takes to place n cuts
  - No branching case (there is just one viable alternative at every step. $O(n)$ is time to compute new X and L:
  - $T(n) = T(n-1) + O(n)$
    - $T(n) = O(n^2)$ Quadratic
  - Branching case (if there are two choices at every step): $T(n) < 2T(n-1) + O(n)$
    - Exponential