De novo Sequence Assembly Framework  
for Next Generation Genome Sequencing

## De Bruijn Graph Construction

K-mer should be hash indexed for immediate lookup. Each k-mer record should include k-mer ID, k-mer itself, coverage, read ID & offset pair list, adjacent list and Vertex ID.

K-mer ID is an int64.

K-mer itself is a string with constant k chars.

Coverage is the occurrence of the k-mer in all reads. It is an int.

Read ID & offset pair list is a list of ID and offset where the k-mer occur in those reads. ID is an integer while a char is enough for offset.

Adjacent list can be represented in 8 bits since there can be at most 4 predecessors and 4 successors. A char is large enough.

Vertex ID is an int64 which will be filled in vertex merge process to indicate the relationship between merged Vertex and k-mer.

Each read can be represented as a list of k-mer. If k-mer IDs are numbered properly, each read can be represented as union of several intervals.

**Construction algorithm:**

First pass:

For each read:

Assign a read ID

For each k-mer in read:

If the k-mer is original:

Create a new entry in hash table

Assign a k-mer ID

Record k-mer string

Coverage = 1

Add read ID and offset to the pair list

Create a reversed complement k-mer entry too

Assign a k-mer -ID

Record reversed complement k-mer string

Coverage = 1

Add read ID and offset to the pair list

Vertex ID = -1

Else:

Coverage++

Add read ID and offset to the pair list

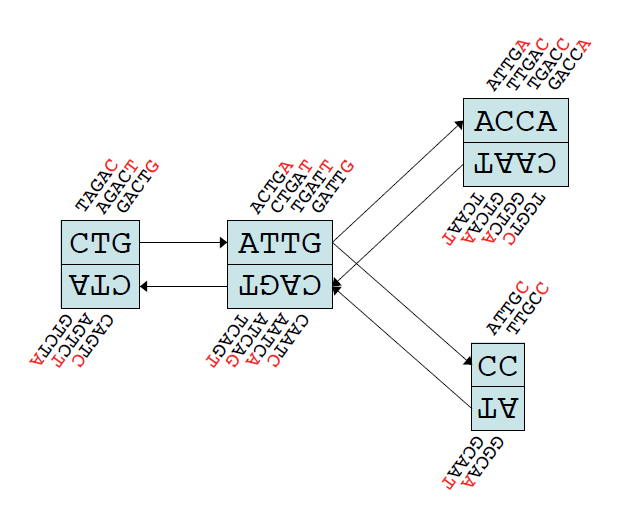
Represent read as k-mer ID interval

Second pass:

For each node

Test 8 possible neighbors, then update their adjacent list and compile adjacent list

## Vertex Merge



**Merge algorithm:**

For each nodeID in AllNodeIDSet

node = GetNodeFromID(nodeID)

if(node != NULL)

MergePossible(node)

**Error correction:**

**Remove tips:**

For each vertex v1:

If v1 has only 1 degree && length of v1 < threshold:

If in degree of v1 = 1:

Find the only predecessor v2

If there exists an out arc of v2 whose multiplicity >= that of the arc to v1:

Remove v1

Do possible merge

Else: // out degree = 1

Find the only successor v2

If there exists an in arc of v2 whose multiplicity >= that of the arc from v1:

Remove v1

Do possible merge

**Remove bubbles:**

Given root

Queue q = empty min heap

q.enque(root)

While q is not empty:

curVertex = q.deque()

For each neiVertex of curVertex:

If neiVertex is not visited:

Record curVertex as the predecessor

neiVertex.dist = curVertex.dist + curVertex.length // may be a different metric

q.enque(neiVertex)

Else:

retVal = false

For each predecessor of neiVertex:

retVal = tryMerge(neiVertex, predecessor, curVertex)

If retVal == true:

break

If retVal == false:

Add curVertex as predecessor of neiVertex

Sub routine tryMerge(endVertex, preV1, preV2):

comVertex = find2Paths(preV1, preV2, P1, P2)

P = Merge(P1, P2)

If P != null:

Update comVertex successor

Update endVertex predecessor

Return true

Else:

Return false

Sub routine find2Paths(preV1, preV2, P1, P2):

Do dijkstra for preV1 and preV2 using their predecessor link

Record the path at each step

Stop when a common vertex is reached

Get 2 paths: comVertex P1 and comVertex P2

Return comVertex

Sub routine merge(P1, P2):

If P1 is similar to P2:

Change the content of P1 or P2 to the one with stronger confidence

Split some vertices of P1 and P2 so that they can be merge at branches

Merge 2 paths

Else:

Return null

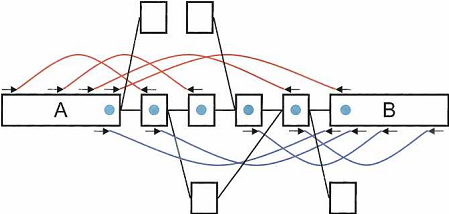
**Remove low coverage paths:**

For each vertex:

If coverage < threshold:

Remove vertex

## Ambiguity resolving



**Algorithm:**

Select seed Vertexs

Use pair information to link seed Vertexs

Find path paving gaps between seed Vertexs

Sub routine select seed Vertexs:

For each Vertex:

If length of Vertex > length threshold (and coverage is less than coverage threshold // ignore it at present)

It is a seed

Sub routine link seed Vertexs:

For each seed Vertexs:

Find seed Vertexs within radius r (Dijkstra)

For each such neighbor Vertexs:

If there are enough pair reads between:

Estimate their distance and link them