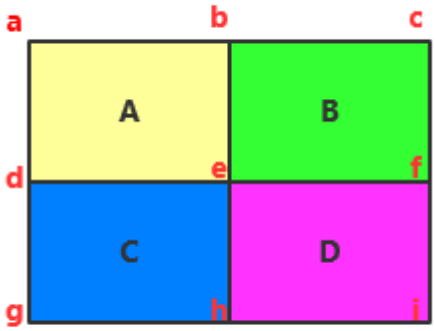
地图调用逻辑

如图中所示，有一个由四个子图（A，B，C，D）组成的大地图，分别讨论在给定下面的任务点的时候，如何 load 子图：



（1）：a → b → e → h

（2）：a → b → c → f → e → h → g

（3）：a → d → g → h → i

对于上述地图，我们对于边的归属，进行分析，得到如下结果：

边	内涵于子图	Tag
(a,b)	A	A
(a,d)	A	A
(b,c)	B	B
(b,e)	A,B	A,B
(c,f)	B	B
(f,e)	B,D	B,D
(f,i)	D	D
(l,h)	D	D
(h,e)	C,D	C,D
(h,g)	C	C
(g,d)	C	C
(d,e)	A,C	A,C

所以，我们在得到上面的边的标签之后，进行下面问题的分析。

（1）：a → b → e → h

我们首先，按照给定的流程格式来逐步分析：

- 读入任务点 a，b，e。
- 对于（a,b）和（b,e）的 tag 取交集，如果有非空的结果，则代表相邻的三个任务点处于同一个子图之中，故，我们 load 该子图。
- 由于（a,b）的 tag 为 A，(b,e)的 tag 为 A，B，则交集为 A，故（a,b）,(b,e)都属于子图 A，所以，load 子图 A，会是后面几项操作的相对最优结果。
- 在 e 点时候，对于(e,h)进行 tag 的分析
- 由于（e,h）的 tag 为 C，D。之前都没有被读取过，故随机选取其中一个子图 load 即

可，这里随机选择 C 子图 load。

- 故，最终的结果为在{a → b → e}load 子图 A，在{e → h}load 子图 C 或子图 D。

（2）：a → b → c → f → e → h → g

- 首先读入 3 个任务点{a,b,c}
- 由于{a → b}和{b → c}的交集为空集，故我们选取{ a → b }的 tag，load 子图 A
- 在 b 点处，读入后面三个任务点{b,c,f}
- 由于{ b → c}和{c → f}的交集为 B，故我们 load 子图 B
- 在 f 点处，读入后面三个任务点{f,e,h}
- 由于{f → e}和{e → h}的交集为 D，故我们 load 子图 D
- 在 h 点处，读入后面的任务点{h,g}
- 由于{g,h}只有一个 tag，所以，我们 load 该子图 C
- 故，最终的结果为在{a → b}处 load 子图 A，在{b → c → f}处 load 子图 B，在{f → e → h}处 load 子图 D，在{h → g}处 load 子图 C。

（3）：a → d → g → h → i

- 在 a 点,读入后面三个点{a,d,g}
- 由于{a → d}和{d → g}的交集为空集，所以，load 子图 A。
- 在 d 点，读入后面三个点{d,g,h}
- 由于{d → g}和{g → h}的交集为 C，故 load 子图 C
- 在 h 点，读入后面的任务点{h,i}
- 由于{h → i}只有一个 tag，所以，load 子图 D
- 故，最终的结果为在{a → d}处 load 子图 A，在{d → g → h}处 load 子图 C，在{h → i}处 load 子图 D。

上面的实例为多子图建图中地图 load 的问题，其逻辑需要遵循以下流程：