```
import { Tree, Switch } from 'antd';
import { CarryOutOutlined } from '@ant-design/icons';

const x = 3;
const y = 2;
const z = 1;
const gData = [];

const generateData = (_level, _preKey, _tns) => {
  const preKey = _preKey || '0';
  const tns = _tns || gData;

  const children = [];
  for (let i = 0; i < x; i++) {
    const key = `${preKey}-${i}`;
    tns.push({ title: key, key, icon: <CarryOutOutlined /> });
    if (i < y) {
      children.push(key);
    }
  }
  if (_level < 0) {
    return tns;
  }
  const level = _level - 1;
  children.forEach((key, index) => {
    tns[index].children = [];
    return generateData(level, key, tns[index].children);
  });
};
generateData(z);

class Demo extends React.Component {
  state = {
    gData,
    expandedKeys: ['0-0', '0-0-0', '0-0-0-0'],
    showLine: true,
    showIcon: true,
    showLeafIcon: true,
  };

  onDragEnter = info => {
    console.log(info);
    // expandedKeys 需要受控时设置
    // this.setState({
    //   expandedKeys: info.expandedKeys,
    // });
  };

  onDrop = info => {
    console.log(info);
    const dropKey = info.node.key;
```

```
const dragKey = info.dragNode.key;
const dropPos = info.node.pos.split('-');
const dropPosition = info.dropPosition - Number(dropPos[dropPos.length - 1]);

const loop = (data, key, callback) => {
  for (let i = 0; i < data.length; i++) {
    if (data[i].key === key) {
      return callback(data[i], i, data);
    }
    if (data[i].children) {
      loop(data[i].children, key, callback);
    }
  }
};
const data = [...this.state.gData];

// Find dragObject
let dragObj;
loop(data, dragKey, (item, index, arr) => {
  arr.splice(index, 1);
  dragObj = item;
});

if (!info.dropToGap) {
  // Drop on the content
  loop(data, dropKey, item => {
    item.children = item.children || [];
    // where to insert 示例添加到尾部，可以是随意位置
    item.children.push(dragObj);
  });
} else if (
  (info.node.props.children || []).length > 0 && // Has children
  info.node.props.expanded && // Is expanded
  dropPosition === 1 // On the bottom gap
) {
  loop(data, dropKey, item => {
    item.children = item.children || [];
    // where to insert 示例添加到头部，可以是随意位置
    item.children.unshift(dragObj);
  });
} else {
  let ar;
  let i;
  loop(data, dropKey, (item, index, arr) => {
    ar = arr;
    i = index;
  });
  if (dropPosition === -1) {
    ar.splice(i, 0, dragObj);
  } else {
    ar.splice(i + 1, 0, dragObj);
  }
```

```
    }

    this.setState({
      gData: data,
    });
  };

  setShowLine = showLine => {
    const { showLeafIcon } = this.state;
    if (showLine) {
      if (showLeafIcon) {
        this.setState({
          showLine: {
            showLeafIcon: true,
          },
        });
      } else {
        this.setState({
          showLine: true,
        });
      }
    } else {
      this.setState({
        showLine: false,
      });
    }
  };

  setShowIcon = showIcon => {
    this.setState({
      showIcon,
    });
  };

  setShowLeafIcon = showLeafIcon => {
    this.setState({
      showLeafIcon,
      showLine: { showLeafIcon },
    });
  };

  render() {
    const { showLine, showIcon, showLeafIcon, expandedKeys } = this.state;
    const { setShowLine, setShowIcon, setShowLeafIcon } = this;
    return (
      <>
        <div style={{ marginBottom: 16 }}>
          showLine: <Switch checked={showLine} onChange={setShowLine} />
          <br />
          <br />
          showIcon: <Switch checked={showIcon} onChange={setShowIcon} />
          <br />
```

```
          <br />
          showLeafIcon: <Switch checked={showLeafIcon} onChange={setShowLeafIcon} />
        </div>
        <Tree
          showLine={showLine}
          showIcon={showIcon}
          className="draggable-tree"
          defaultExpandedKeys={expandedKeys}
          draggable
          blockNode
          onDragEnter={this.onDragEnter}
          onDrop={this.onDrop}
          treeData={this.state.gData}
        />
      </>
    );
  }
}

export default () => <Demo />;
```